A COOPERATIVE APPROACH TO CONTINUOUS CALIBRATION IN MULTI-PROJECTOR DISPLAYS

Tyler M. Johnson

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2009

Approved by:

Henry Fuchs

Anselmo Lastra

Marc Pollefeys

Jack Snoeyink

Herman Towles

Greg Welch

ABSTRACT

TYLER M. JOHNSON: A Cooperative Approach to Continuous Calibration in Multi-Projector Displays
(Under the direction of Henry Fuchs)

Projection-based displays are often used to create large, immersive environments for virtual reality (VR), simulation, and training. These displays have become increasingly widespread due to the decreasing cost of projectors and advances in calibration and rendering that allow the images of multiple projectors to be registered together accurately on complex display surfaces, while simultaneously compensating for the surface shape—a process that requires knowledge of the pose of each projector as well as a geometric representation of the display surface.

A common limitation of many techniques used to calibrate multi-projector displays is that they cannot be applied continuously, as the display is being used, to ensure projected imagery remains precisely registered on a potentially complex display surface. This lack of any continuous correction or refinement means that if a projector is moved slightly out of alignment, either suddenly or slowly over time, use of the display must be interrupted, possibly for many minutes at a time, while system components are recalibrated.

This dissertation proposes a novel framework for continuous calibration where "intelligent" projector units (projectors augmented with cameras and dedicated computers) interact cooperatively as peers to continuously estimate display parameters during system operation. Using a Kalman filter to fuse local camera image measurements with remote measurements obtained by other projector units, the projector units in a multi-projector display cooperatively estimate the poses of all projectors and information about the display surface, such as its pose or shape, in a continuous fashion. This decentralized approach to continuous calibration has the potential to enable scalable and fault-tolerant display solutions that can be configured and maintained by untrained users.

To My Great-Uncle Richard H. Lien,

A Gentleman and Scholar,

A Man of Great Determination

ACKNOWLEDGMENTS

I would like to thank my advisor, Henry Fuchs, for introducing me to the world of multi-projector displays and continuous calibration and for keeping me motivated with his great enthusiasm for the subject. This work incorporates but a fraction of the many wonderful ideas Henry has shared with me throughout the course of my studies.

I would also like to thank the other members of my committee: Anselmo Lastra, Marc Pollefeys, Jack Snoeyink, Herman Towles (reader), and Greg Welch (reader). This dissertation incorporates, and has been greatly improved by, all of the advice, insightful comments, and guidance they have given me. I would especially like to thank Greg Welch for his idea of passing state parameters between the projector units and for discussions that helped in formulating the filter described in this dissertation.

I also wish to thank all of the many students and collaborators that I have had the pleasure of working with throughout my time in graduate school: Patrick Quirk, Rick Skarbez, Florian Gyarfas, Eric la Force, Stephen Guy, Brendan Walters, Fu-Che Wu, Gu Ye, Andrei State, Brad Colbert, Brian Clipp, David Gallup, Adrian Ilie, John Hansen, David Feng, Jeremy Wendt, and Mary Whitton. In many cases, this work has benefitted directly or indirectly as a result of their efforts. I would also like to recognize John Thomas for his help in many aspects of this work including the construction of the experimental spaces and model house.

I am also grateful to my family for all of their encouragement and support. I thank my parents, Michael and Jeanne, for teaching me the importance of education, pride in my work, and for giving me the skills and motivation to succeed. They have always been a great source of inspiration to me. This dissertation is dedicated to my great-uncle Richard, who showed me that all goals can be achieved with enough determination. Finally, I would like to thank my wife, Emily, for her love and support, and for making this work possible by leaving everything behind to join me in the United States.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
BRDF	Bidirectional Reflectance Distribution Function
COP	Center of Projection
CPU	Central Processing Unit
DLP	Digital Light Processing
DLT	Direct Linear Transform
DOF	Degrees of Freedom
FOV	Field of View
GPU	Graphics Processing Unit
IPU	Intelligent Projector Unit
KLT	Kanade-Lucas-Tomasi Feature Tracker
LOF	List of Figures
LOT	List of Tables
OpenCV	Intel Open Source Computer Vision Library
PC	Personal Computer
RANSAC	RANdom SAmple Consensus
SCAAT	Single-Constraint-at-a-Time
SLAM	Simultaneous Localization and Mapping
TOC	Table of Contents
VR	Virtual Reality

CHAPTER 1

INTRODUCTION

The advent of the affordable commodity projector and the ease with which this device can be connected to a source of computer generated imagery has created a surge of research interest in the projector's unique ability to turn almost any physical surface into a display. From office workspaces to virtual environments, projection-based displays have proven to be viable alternatives to previously established display technologies, while offering their own unique advantages. The ability to combine multiple projectors into a single seamless display without the unsightly borders of other display technology makes the projector ideal for creating large, immersive environments for virtual reality (VR), simulation, and training. Projection-based displays can also be quite flexible, since the size and field-of-view of the display can be adjusted to suit a variety of different application requirements simply by rearranging or repositioning the projectors.

Unfortunately, most projection-based displays in use today cannot take advantage of this flexibility due to a reliance on tedious and time-consuming manual calibration and alignment procedures that must be repeated at even the slightest change in display configuration, such as the movement of a projector. To address this, algorithms have recently been developed that can automatically adjust to such changes during actual display use. Such so-called *continuous calibration* approaches, which are the subject of active research, are the topic of this dissertation and have the potential to dramatically improve the usefulness and reliability of projection-based displays. Before delving deeper into this topic, however, an introduction to the general problem of calibration and image correction in multi-projector displays is provided.

1.1 Geometric Considerations

As the reader will observe over the course of this work, many of the challenges involved in projection-based displays are *geometric* in nature. For example, when projecting onto a surface that is complex (non-flat), such as the room corner in Figure 1.1 left, the imagery will bend and distort around the surface according to its shape. This often results in a display that is unusable since the observed imagery is not consistent with what the viewer is intended to observe.

Figure 1.1: Left: A simple room corner used for projection-based display without correction for the shape of the display surface. Right: The same display after applying basic techniques to correct for the display surface shape.

Fortunately, by appropriately distorting the imagery *before* projection, it is possible to neutralize this effect for a given viewing location, causing the imagery to appear without distortion, as it was originally intended (Figure 1.1 right). As will be described in more detail in Chapter 3, the ability to pre-distort the imagery in this way requires that the following information is known:

- The shape of the display surface
- The relative position and orientation of the projectors
- The viewing location

These parameters must be known to a high degree of accuracy in order for image correction to be successful. This is especially important for displays with multiple overlapping projectors, such as that of Figure 1.2, since errors in these parameters can result in image misregistration or misalignment

that is revealed as blurring or "ghosting" in the imagery. For this reason, it is typically desired that the above parameters are known to such a high degree of accuracy that sub-pixel registration between overlapping projected images is achieved.

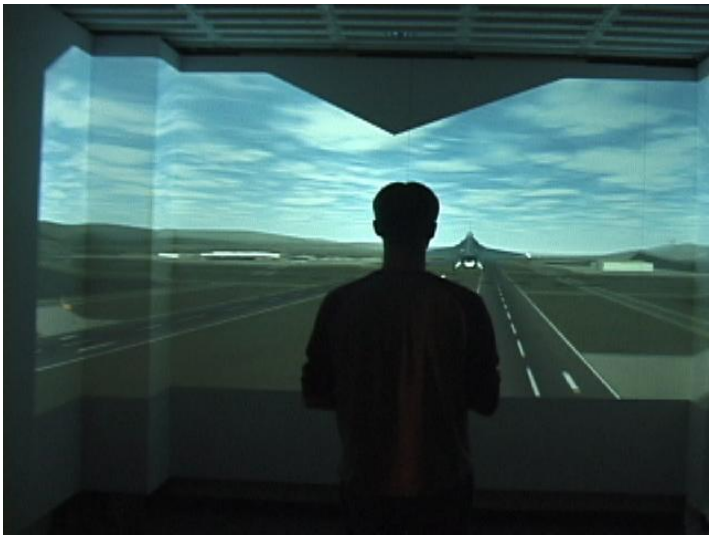


Figure 1.2: A room-sized display using two overlapping projectors.

1.2 Photometric Considerations

In addition to the aforementioned geometric challenges, there are also various *photometric* factors to consider that may greatly affect the quality of a projection-based display. While geometric correction is concerned with placing each pixel in its proper place on the surface, photometric correction is concerned with ensuring each pixel has the proper color and brightness to produce a display with no visible seams that would delineate the imagery of multiple projectors. For example, in areas where imagery from multiple projectors overlaps on the surface, as in Figure 1.3, a brighter than normal region will be visible due to the light contribution from multiple projectors being greater than that of a single projector. Other photometric issues include compensating for the color of the display surface, compensating for differences between projectors in color gamut and input-output response, and accounting for the effects of light scattering between different parts of the display surface.

Just as it is possible to correct for the undesirable geometric effects in a multi-projector display, the various photometric considerations that have been mentioned can be compensated for as well.

Areas of increased brightness can have their image brightness attenuated to *blend* the images from separate projectors together. Correction for differences in color-gamut and input-output response can be achieved via altering the color and brightness of pixels before projection so that the resulting color and brightness on the surface is matched across projectors. When properly combined with geometric correction, it is possible to achieve a truly seamless multi-projector display.

Figure 1.3: A two-projector display without and with photometric blending. Notice the brighter area in the center of the left image.

1.3 Continuous Calibration

In general, correcting for the potential geometric and photometric issues in a projection-based display can be cast as the following problem: given a desired image that the user should observe, how should this image be warped (its pixel rearranged) and its color values altered, so that the imagery from all projectors combines together on the display surface to produce a coherent, seamless image. This process is generally referred to as *image correction*. In most projection-based displays, the desired image is rendered by a graphics application. In the case of an interactive application, such as a flight simulator, the desired image will change from frame to frame with any motion of the scene, requiring a real-time geometric and photometric correction process for projected imagery.

The integration of geometric and photometric correction into an interactive application is illustrated in Figure 1.4. As each frame is rendered by the application, geometric and photometric correction is performed before the image is sent to the projector for display. As illustrated in the figure, this

is an ongoing, repetitive process that occurs as the display is being used.

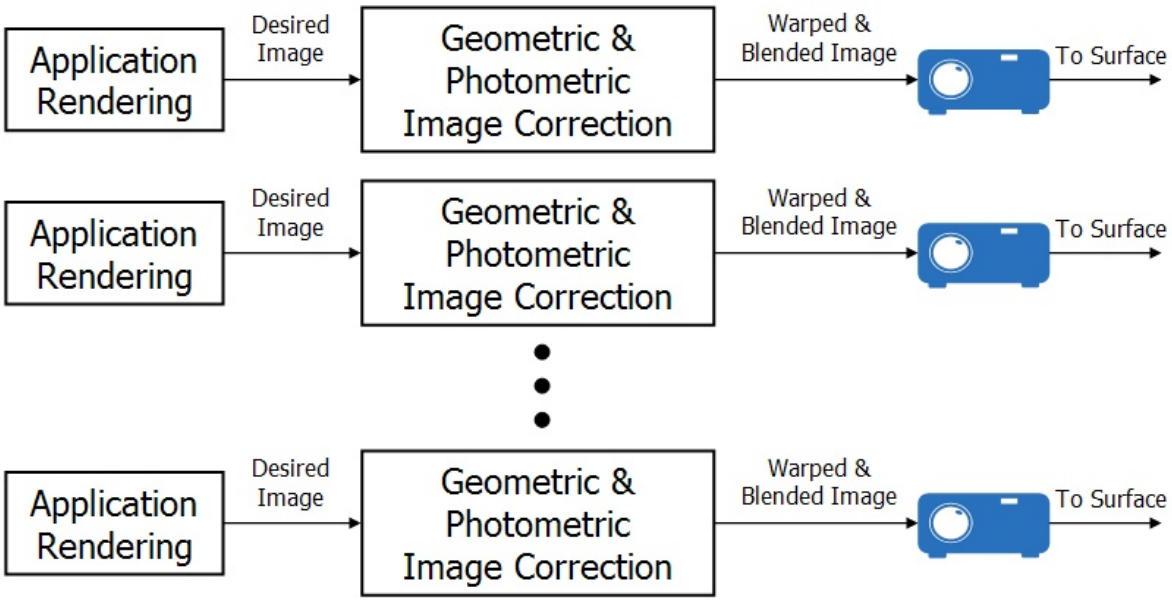


Figure 1.4: Geometric and photometric image correction is performed on images rendered by the application before they are sent to the projector for display.

As mentioned previously, the ability to perform geometric and photometric correction requires certain information about the display configuration to be known, for example, the shape of the display surface, the relative position and orientation of the projectors, and/or the input-output response of the projector(s). Currently, most techniques for geometric and photometric correction require an *upfront* calibration process that occurs before the display is used and assumes a fixed display configuration where display parameters do not change. This inability to adapt to changes in the display is a limitation since even for "fixed" display configurations, display parameters are likely to change over time due to inadvertent physical perturbations caused by vibration of the mounting structure, heating and cooling cycles, and even gravity. One can also imagine situations where the user may wish to deliberately alter the display configuration due to varying field-of-view or spatial resolution requirements between applications. Displays that are not able to adapt to these changes automatically require periodic manual recalibration in order to maintain a consistent level of display quality over time. This can be costly; in many commercially available systems, recalibration must be performed on-site by specially-trained professionals.

For these reasons it is desirable for a multi-projector display to continuously monitor and adapt

to changes in display parameters over time as the display is being used. This process of adapting to display changes during actual display use is referred to as *continuous calibration*. As illustrated in Figure 1.5, continuous calibration is implemented as a module that runs concurrently with image correction and rendering, providing estimates of the latest display parameters to the correction software. Some of the enabling technologies behind continuous calibration include the programmable graphics processing unit (GPU), which more easily accommodates changing display parameters than earlier static image correction solutions, and affordable commodity digital cameras that form the underlying sensor system in many continuous calibration approaches.

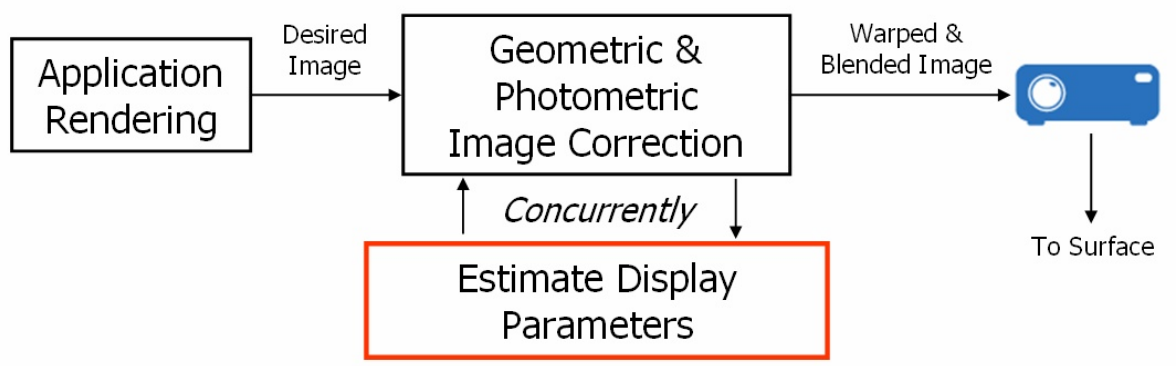


Figure 1.5: Continuous calibration occurs as the display is being used and affects the geometric and photometric correction that occurs.

1.4 Contributions

This dissertation presents several novel techniques for upfront and continuous calibration designed to improve the utility of the projection-based displays. To begin with, the use of projectors equipped with wide-angle lenses is examined. Such projectors can have an advantage over traditional projectors in creating immersive display environments since they can be placed in close proximity to the display surface to reduce user shadowing issues while still producing large images. However, wide-angle projectors exhibit severe image distortion requiring the image generator to correctively pre-distort the output image.

This dissertation describes a new technique based on [Raskar et al., 1998]'s two-pass rendering algorithm that is able to correct for both arbitrary display surface geometry and the extreme lens

distortion caused by fisheye lenses. Methods for implementing this distortion correction algorithm in a real-time shader program running on a commodity GPU to create low-cost, personal surround environments are also described.

Continuous calibration approaches must be scalable in order to be useful in large displays with many projectors. In this respect, a distributed calibration methodology is better suited and can provide greater fault tolerance than centralized approaches, where all calibration data is aggregated and processed on a single machine. This dissertation also proposes a novel *distributed cooperative framework* for continuous calibration in multi-projector displays where "intelligent" projector units (projectors augmented with cameras and dedicated computers) interact cooperatively as peers to continuously estimate display parameters over time.

In this framework, each projector unit continuously measures the location of features in the projected imagery using its cameras. These local measurements are fused with remote measurements obtained by its peers using a Kalman filter. It is shown that this framework can be used during display to estimate the poses of all projectors in a multi-projector display, as well as information about the display surface, such as its shape or pose.

1.5 Thesis Statement

The central thesis of this dissertation is:

Geometric calibration of a multi-projector display can be maintained continuously, during system operation, using a distributed cooperative approach that simultaneously refines

I. *the poses of multiple projectors and*

II. *the geometry of the display surface.*

1.6 Additional Contributions

In addition to this distributed cooperative framework, this dissertation makes several other contributions to the area of calibration in multi-projector displays. In addition to self-contained "intelligent"

projector units, one can also imagine continuously calibrated displays where the cameras may be separate from the projector(s). This idea is explored in published work [Johnson and Fuchs, 2007a] included at the end of this document in the Appendix. Another published work [Johnson and Fuchs, 2007b] included in the Appendix explores the use of projector-camera systems in the office and describes how the task of displaying onto the surfaces of an office cubicle by combining projection and flat-screen technology can be unified with the task of providing camera-based desktop scanning.

1.7 Publications

This dissertation is based the following published works that have appeared in the proceedings of peer-reviewed conference proceedings, in book chapters, and in workshops:

T. Johnson, G. Welch, H. Fuchs, E. La Force, H. Towles, A Distributed Cooperative Framework for Continuous Multi-Projector Pose Estimation, In *Proceedings of the 2009 IEEE Virtual Reality Conference*, Lafayette, LA, March, 2009.

H. Towles, T. Johnson, H. Fuchs, Projector Based Displays, *The PSI Handbook of Virtual Environments for Training and Education*. Vol 2 Ed. Denise Nicholson, Dylan Schmorrow, and Joseph Cohn. Westport: Praeger Security International, 2009. 63-89.

T. Johnson, H. Fuchs, A Unified Multi-Surface, Multi-Resolution Workspace with Camera-Based Scanning and Projector-Based Illumination, In *Eurographics Symposium on Virtual Environments/Immersive Projection Technology Workshop 2007*, Weimar, Germany, July, 2007.

T. Johnson, H. Fuchs, Real-Time Projector Tracking on Complex Geometry Using Ordinary Imagery, In *4th IEEE International Workshop on Projector-Camera Systems (ProCams 2007)*, Minneapolis, MN, June, 2007.

T. Johnson, F. Gyarfas, R. Skarbez, H. Towles, H. Fuchs, A Personal Surround Environment: Projective Display with Correction for Display Surface Geometry and Extreme Lens Distortion, In *Proceedings of the 2007 IEEE Virtual Reality Conference*, Charlotte, NC, Mar, 2007.

P. Quirk, T. Johnson, R. Skarbez, H. Towles, F. Gyarfas, H. Fuchs, RANSAC-Assisted Display Model Reconstruction for Projective Display, In *Proceedings Emerging Display Technologies 2006*, Alexandria, VA, Mar, 2006.

1.8 Outline

This dissertation is organized as follows.

- Chapter 2 describes previous work related to geometric calibration in projection-based displays. Special care is taken in this section to differentiate techniques based on their ability to provide some type of automatic or continuous calibration.
- Chapter 3 introduces some preliminary concepts and mathematics that will prove useful to the reader in understanding the techniques and concepts presented in this thesis.
- Chapter 4 represents the first contribution of this work, which is a novel technique for calibration and image correction in projection-based displays that include a significant amount of projector lens distortion.
- Chapter 5 introduces the distributed cooperative framework for continuous calibration in multi-projector displays by examining the operation of the framework in the special case that the geometry of the display surface is static and known prior to system operation and only the poses of the projectors must be estimated.
- Chapter 6 describes an extension to the framework of the previous chapter that provides the ability to simultaneously estimate information about the display surface and the poses of the projectors.
- Chapter 7 provides a summary and a discussion of future work.
- The Appendix contains two additional publications whose contents have not been included in the body of the dissertation.

CHAPTER 2

PREVIOUS WORK

The concept of using imagery to create a desired optical effect from a particular viewpoint has been in existence since the Renaissance. Using a technique called *trompe-l'œil*, artists were successful in fooling the eye with illusionistic paintings that could make a hallway appear to continue through a wall or a ceiling appear to possess an open portal to the sky. This was made possible by the artists' advanced knowledge of perspective, which allowed them to create art rendered from true vanishing point perspective. In much the same way, a well-calibrated multi-projector display creates, for example, the illusion of a flat display surface even though its true geometry may be much more complex.

This chapter describes previous work in multi-projector displays beginning with a review of the first techniques that were developed for *camera-based calibration*: those requiring a manual upfront calibration process. This is followed by a discussion of several approaches designed to automate the calibration process in a variety of scenarios. Continuous techniques designed to calibrate the display during operation are then introduced before concluding with a discussion of previous work in the related area of distributed Simultaneous Localization and Mapping (SLAM).

2.1 Upfront Techniques

The ability to determine how a desired image should be warped to appear correct to a viewer in a multi-projector display has traditionally relied on an upfront calibration process where cameras are used to measure properties of the display configuration such as the geometry of the display surface and the position and orientation of each projector. A brief summary of upfront calibration techniques

follows.

[Surati, 1999] proposed a digital approach to calibrating tiled multi-projector displays on a *planar* surface that did not require the manual projector alignment of previous techniques. The basis for this approach is to use a static camera to observe patterns projected by each projector, allowing a pixel-to-pixel mapping to be established between the camera and projectors. By additionally imaging a physical grid pattern placed on the display surface to establish the screen space, a second mapping between the camera pixels and the screen is established. The mappings from camera-to-screen-space and from projector-to-camera can then be composed to achieve geometric registration between multiple projectors.

Concurrent work [Raskar et al., 1998, Raskar et al., 1999a] showed how camera-based calibration could be expanded to compensate for arbitrary surfaces. This work used time-multiplexed coded structured light patterns, first proposed by [Posdamer and Altschuler, 1982], to establish image correspondences between projectors and a set of calibrated cameras. The calibration and correspondence information allows a point-sampled representation of the display surface model to be reconstructed, which is then triangulated into a polygonal mesh for rendering. The set of 3D-2D correspondences in each projector resulting from the reconstruction is also used to calibrate each projector using the Direct Linear Transform (DLT) algorithm [Abdel-Aziz and Karara, 1971]. At display time, the display surface model and projector calibration are used in conjunction with knowledge of the user's position to compensate for the geometry of the display surface using two-pass rendering, an algorithm that will be discussed further in the next chapter.

[Bimber et al., 2005] compensates for complex display surface geometry by directly measuring the warping functions that are induced by the display surface between the viewer and each projector. This is done by placing cameras at various unstructured locations around the environment and measuring the pixel-to-pixel mapping between these cameras and the projectors. To compensate for the geometry of the display surface at run-time for arbitrary viewpoints, the measured per pixel mappings are interpolated using a technique similar to unstructured lumigraphs [Buehler et al., 2001] that takes into account the position of the viewer with respect to the known camera locations where the mappings were measured.

2.2 Automatic Upfront Techniques

Some advanced upfront calibration techniques are able to calibrate automatically without the need for manual input from the user. Such techniques are the topic of this section. A common limitation of both upfront and automatic calibration techniques, however, is that they are sensitive to even small changes in display configuration, such as a projector being bumped slightly out of alignment. In such an event, use of the display must be interrupted, potentially for many minutes at a time, while the system is recalibrated.

The PixelFlex system [Yang et al., 2001, Raij et al., 2003] featured mirrors mounted on pan-tilt units that allowed each projector's image to be easily and quickly repositioned to create new display configurations. Each new configuration could be automatically calibrated within a matter of minutes by automating the process of mapping projector pixels to screen pixels using a dedicated camera as described in [Surati, 1999]. The PixelFlex system also incorporated a number of advancements in photometric correction used to achieve seamlessness across multiple projectors [Majumder, 2002, Majumder and Stevens, 2002, Majumder et al., 2000].

[Chen et al., 2002] describes a vision-based system for automatically calibrating tiled multi-projector displays. By observing the projection of calibration images with a number of camera views sufficient to cover the entire displayable surface, the idea of a camera homography tree is introduced to link each camera and projector to the global reference frame through a chain of homographies. Methods of optimizing the homography tree and estimating both local and global alignment error are also presented.

In [Raskar et al., 2003] a method is described for automatically calibrating a cluster of projectors acting as a tiled display on a planar or quadric display surface. This work also demonstrated the ability to easily incorporate new projectors into an already calibrated display. Each projector is equipped with its own attached camera, and the two devices are pre-calibrated to one another. When a new projector joins the display, projection of user imagery is interrupted by a calibration procedure where projectors alternately display calibration images used to compute pair-wise homographies or quadric transfer parameters. The pair-wise homographies or quadric transfer parameters are then globally adjusted to reduce error. In the case of a planar surface, each projector independently computes the largest

inscribed rectangle residing within the union of all projectors' extents on the display surface. This rectangle is used to establish the coordinate system of the display on the wall.

[Raij and Pollefeys, 2004] showed that given a camera with known intrinsics, it is possible to automatically calibrate a multi-projector display on a planar display surface. Constraints on the calibration are produced by using structured light to generate correspondences between each projector and the camera. After correspondences have been found, the intrinsics of the projectors as well as the extrinsics of each projector and the camera with respect to the display surface are calculated using a linear system followed by a non-linear refinement that assumes all projectors have the same unknown principal point. This is a departure from previous techniques that were either homography-based and lacked a full 3D calibration or used the DLT algorithm [Abdel-Aziz and Karara, 1971] for projector calibration.

[Bhasker et al., 2006] proposed an asynchronous approach to calibration and photometric correction in multi-projector displays. While limited to a planar display surface and a tiled display configuration, the work is truly scalable to a large number of projectors since calibration and rendering are not performed in a centralized way. By projecting patterns in an upfront calibration step, each projector is able to automatically and independently determine its location in the tiled display and compute the geometric and photometric corrections it must perform to create a seamless display.

2.3 Continuous Techniques

In general, the continuous calibration techniques that have been developed for projection-based displays can be divided into two categories: *active* and *passive*. In the first are techniques where calibration aids that are imperceptible to a human observer are injected into the user imagery. These techniques are called active since they affect what is displayed by the device. Techniques that make no change to the imagery displayed by the projector are called passive and instead rely on their ability to extract calibration information from the application imagery.

[Cotting et al., 2004] describes an active technique that is able to automatically and continuously estimate the geometry of the surface being used for projection. This is accomplished by taking advantage of the mirror flip sequences used in DLP® projectors to embed imperceptible calibration images

into the application imagery. Embedding is done by modifying the projected image slightly so that a synchronized camera exposed during a short time slice of the total frame duration will observe an embedded pattern. By embedding coded structured light into the images projected by the user, surface depth estimates can be obtained each frame and used to update the model of the display surface used in rendering. While the size of the time slice can be chosen small enough to make the embedded pattern imperceptible to the user, the embedding process can result in a shift in the color values of the displayed imagery.

[Cotting et al., 2005] improves upon the original result by describing how interference of the encoded signals when using multiple projectors can be avoided to create large displays using imperceptible structured light by multiplexing in time. The work also describes how the displacement of projector-camera modules during use of the display can be compensated for by reconstructing the pattern projected by the displaced projector using cameras on other modules.

[Grundhöfer et al., 2007] also describes an active embedding technique. Their approach is to modify projected images at each pixel by some Δ amount that can be detected by an observing camera. Using a stereo projector, the modified image and its complement are projected as the left- and right-eye images of the stereo projection. Since the images are projected time sequentially at 60 Hz, the images integrate imperceptibly to form the original image. Synchronized cameras with shared optical axes are able to capture the left- and right-eye images separately, allowing the coded pattern to be reconstructed by taking the ratio of the two images. The authors note that careful choice of the Δ parameter is needed at each pixel to prevent perception of the pattern in certain usage scenarios.

[Yang and Welch, 2001] is the first work to demonstrate the ability of a projector-camera system to passively estimate the geometry of a display surface during display-time using only features present in application imagery. The estimate of the surface begins as a single plane, and as features between the projector and camera images are matched, the surface estimate is continuously refined over time. An independent Kalman filter at each projector pixel processes the feature measurements that fall on that pixel and updates the pixel's depth estimate. This filter also provides an assessment of the uncertainty in each estimate as well as a certain robustness against false correspondences.

[Zhou et al., 2008] also describes a passive system for continuous calibration that forgoes the introduction of artificial features in favor of calibration using features already present in the imagery

projected by the user. In a multi-projector display with known, complex display surface geometry, this system demonstrated the ability to recalibrate projectors that have been moved using knowledge of the surface shape and the calibration of other projectors that are known not to have moved. In cases where the display surface is planar and the projectors are static, the system can also compensate for motion of the display surface.

[Gupta and Jaynes, 2006] describes an interactive display where the pages of blank book are augmented with projected multi-media content such as images and video. The projected imagery is automatically and continuously registered to the pages of the book as they are turned by measuring feature correspondences in the projected imagery between the projector and a calibrated camera. The authors also demonstrate the ability to navigate volumetric datasets by removing a page from the book.

[Zollmann et al., 2006] proposes a hybrid projector display that is able to calibrate continuously using both passive and active techniques. Correction for complex surface geometry is performed in a manner similar to [Bimber et al., 2005] except the user is required to wear a head-mounted camera that observes the projected imagery. Through matching features in the application imagery, the system is able to make minor adjustments to the calibration in a passive way. When the system determines the errors are too large to correct passively, the application imagery is interrupted while an active calibration process visible to the user occurs to bring the display back into calibration.

2.4 Distributed SLAM Techniques

In the most general sense, a continuous calibration algorithm will attempt to estimate the full calibration of all projectors, the geometry of the display surface, and the location of the viewer in a common coordinate system. The challenge of continuous calibration in a multi-projector display thus shares some similarity to the Simultaneous Localization and Mapping (SLAM) problem, which refers to the problem of generating a map of an unknown environment using a mobile vehicle or robot whose pose in the environment must also be estimated. For this reason, a brief discussion of previous work in the area of SLAM is included, with a strong focus on *distributed* SLAM, the problem of localizing multiple robots in a distributed fashion, which is most similar to the continuous projector display calibration approach described in this dissertation.

The seminal work in vision-based SLAM was carried out in the late 1980s by [Smith et al., 1989], who provided an elegant solution to the problem through the use of an extended Kalman filter whose state consisted of both the vehicle pose and map estimates.

[Walter and Leonard, 2004] describes a solution to distributed SLAM in the case of a heterogeneous mix of robots where so-called *masters* have the ability to self-navigate using SLAM while *slaves* have a more limited set of sensors for dead-reckoning and measuring the environment. Using moving baseline navigation, inter-vehicle and feature observations made by different robots are integrated by the *master* vehicle via a Kalman filter.

[Dellaert et al., 2005] formulates the problem of distributed SLAM as a large-scale inference problem on a factor graph and describes how it can be solved using a distributed Multi-Frontal QR decomposition. In this work, one vehicle is designated to be the coordinating node with which all other vehicles communicate. A *clique tree* [Pothen and Sun, 1992, Blair and Peyton, 1993] or *junction tree* [Cowell et al., 1999] that is consistent with this topology is then computed on the unknowns such that the QR computation can be distributed among the vehicles in an efficient manner.

CHAPTER 3

PRELIMINARIES

This chapter is intended to provide a brief introduction to techniques and concepts that will be referred to in later chapters. It begins with a discussion of commonly used geometric models for projectors and cameras and describes how these are related to geometric image correction in multi-projector displays. This information will prove useful in Chapter 4 when developing methods for correcting wide-angle lens distortion of projectors. Some results from the study of two-view geometry that provide useful constraints for calibration are then provided before concluding with a discussion of various techniques for estimation that will be used in formulating the distributed cooperative framework for continuous calibration introduced in Chapter 5.

3.1 Geometric Models for Projectors and Cameras

Geometric models for cameras and projectors are needed for a variety of tasks. For example, we may wish to determine the 3D location of a point in the world given measurements of the 2D point in images captured by two different cameras. This is a common problem in reconstructing the structure of a scene that is viewed by multiple cameras and requires a geometric model for the cameras that includes such parameters as the position and orientation of the cameras in the world and the focal length of the camera lenses.

While cameras and projectors serve different purposes, both use a lens to direct light. Cameras use a lens to focus *incoming* light onto a sensor, while projectors use a lens to focus *outgoing* light onto a surface. This symmetry means that cameras and projectors can be thought of as optical duals

of one another, and geometric models for cameras can be applied to projectors without modification. Some of the more widely-used geometric models for projectors and cameras will now be discussed.

3.1.1 Pinhole Perspective Model

The most common lens model for cameras and projectors is the *pinhole perspective model*, which describes the common pinhole camera. In this model, straight lines in the world correspond to straight lines in the image as they pass through the lens. The pinhole perspective model includes 11 degrees of freedom that are represented as a 3×4 projection matrix P, defined up to scale, that relates a 3D world point $X = [X, Y, Z, 1]^T$ to its 2D imaged pixel location $x = [u, v, s]^T$ through the linear equation

$$x = PX. \tag{3.1}$$

The 11 degrees of freedom afforded by the pinhole perspective model can be divided into *intrinsic* and *extrinsic* parameters. The extrinsic parameters include six degrees of freedom for the position and orientation of the device in world coordinates. The orientation is represented as a 3×3 rotation matrix R with three degrees of freedom and the position is represented by the 3-vector C. The intrinsics describe the focal properties of the lens and remain constant as the device is moved around in the world. The intrinsics include 5 parameters: the focal length in u and v image coordinates (f_u, f_v), the location of the principal point in the image (p_u, p_v) and a skew parameter s used to account for non-square pixels. The intrinsics are represented as a 3×3 matrix K with

$$K = \begin{bmatrix} f_u & s & p_u \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

These intrinsic and extrinsic parameters are combined together to form the projection matrix P of the camera from Equation 3.1 in the following way

$$P = K[R|-RC]. \tag{3.3}$$

The rotation matrix R and translation vector $-RC$ together describe the transformation of the

world point X into the coordinate system of the device. The K matrix transforms points on the image plane to pixel coordinates in the camera or projector image by applying the focal properties of the lens.

Lens Distortion

While lens properties can vary drastically, no physical lens is a perfect pinhole and straight lines in the world imaged by a camera, or in an image projected by a projector, will be distorted to a greater or lesser extent by the lens. In practice, cameras and projectors are often treated as pinhole devices along with a distortion model that attempts to compensate for deviations from the ideal pinhole model.

Since lens distortion affects where world points are imaged (or in the case of projectors, the direction of the ray along which a pixel is projected), estimation of lens distortion in a device can significantly improve calibration results, especially for wide-angle lenses. The most important type of distortion is radial, which increases with distance from the center of distortion in the image. The center of distortion is usually located at or near the principal point. In general, the amount of radial distortion is inversely proportional to the focal length of the lens.

In the computer vision literature [Hartley and Zisserman, 2003, Heikkil and Silvn, 1997], camera lens distortion is typically modeled as a process that occurs after the projection of a world point onto the image plane of the camera. Using the decomposition of $P = K[R|t]$, where R is the camera's rotation matrix, and $t = -RC$ with C the camera's center of projection, distortion is incorporated into the projection process as

$$x = K\delta_{in}([R|t]X). \tag{3.4}$$

The δ_{in} operator remaps homogeneous 2D points after their initial projection onto the image plane to model deviations from the pinhole model for light *entering* the lens. This operator will necessarily be non-linear in order to capture non-linearities in the projection process not modeled by the pinhole projection model of Equation 3.1. Since each homogeneous 2D point on the image plane is the projective equivalent of a ray, this function can also be thought of as operating on rays as they enter the lens. An image plane example of a δ_{in} operator distorting an image border is depicted in Figure

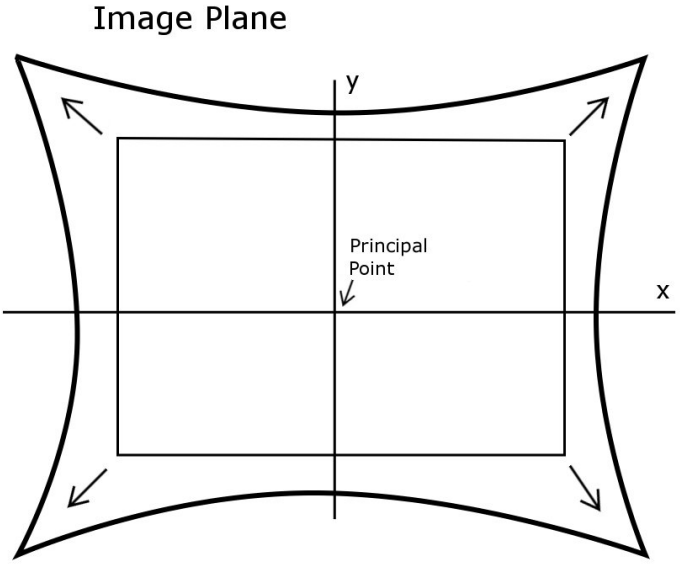


Figure 3.1: Example effect of lens distortion (pincushion) on the image plane.

3.1. The operation depicted in the figure is referred to as a pincushion distortion.

A technique developed by Brown [Brown, 1971], which has been adopted by the Matlab Camera Calibration Toolbox and Intel's OpenCV, models both radial and decentering lens distortion, where the latter arises from imperfectly aligned lenses. Decentering distortion possesses both radial and tangential components. For the Brown model with two coefficients for radial distortion (k_1, k_2) and two for tangential (p_1, p_2), the distortion operator δ_{in} is

$$\delta([u,v,s]^T) = \begin{bmatrix} a(1+k_1r^2+k_2r^4)+2p_1ab+p_2(r^2+2a^2) \\ b(1+k_1r^2+k_2r^4)+p_1(r^2+2b^2)+2p_2ab \\ 1 \end{bmatrix}, \tag{3.5}$$

where $a = u/s$, $b = v/s$ and $r^2 = a^2 + b^2$.

While this model is not directly invertible, the set of coefficients that invert a specific distortion can be determined using a non-linear optimization based on correcting a set of distorted sample points in the image plane of the device to their original positions [Heikkil and Silvn, 1997].

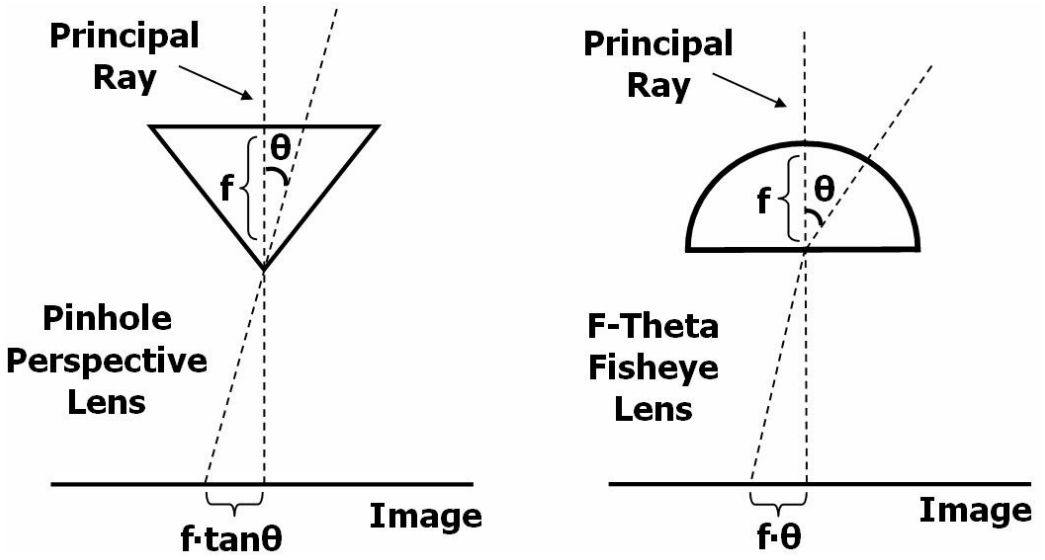


Figure 3.2: Pinhole perspective versus f-theta fisheye lens.

3.1.2 Non-Pinhole Lens Models

Some lenses represent such a strong departure from the pinhole lens model that they require a completely different representation. For example, a fisheye lens can have a field of view as great as 180°. In the pinhole perspective model, the angle θ in radians between an incoming ray and the principal ray is related to the focal length f in pixels and the distance r between the principal point and the incoming ray's pixel by $r = f\tan\theta$. While there are a number of different fisheye lenses with unique properties, consider the most common type, an equidistance projection or f-theta lens where $r = f\theta$. The relationship between a pinhole perspective lens and an f-theta fisheye lens is depicted in Figure 3.2.

Non-pinhole lens models can be dealt with in the same way as the pinhole model - as functions operating on homogeneous 2D points on the image plane. For example, given this framework, the δ_{in} for an f-theta fisheye lens can be expressed as

$$\delta_{in}([u,v,s]^T) = \begin{bmatrix} \arctan(r) * x \\ \arctan(r) * y \\ r \end{bmatrix}, \tag{3.6}$$

where $x = u/s$, $y = v/s$ and $r = \sqrt{x^2 + y^2}$. The focal length has been omitted from the above equation

since it is equal to one on the image plane before the intrinsic matrix is applied. This can be thought of as treating the fisheye lens as a pinhole lens with a distortion function that models its deviation from the pinhole model. Of course, it is possible in practice that the behavior of the fisheye lens may deviate somewhat from its own ideal model for the same reasons that a pinhole lens might. These additional "distortions" can be modeled separately and incorporated into the δ_{in} or δ_{out} function of the lens model.

In this example, δ_{in} can be directly inverted to produce δ_{out}, the distortion that light undergoes as it exits the device. Also note that since the f-theta model is treated as a distortion within the pinhole model, a singularity exists for $\theta = \pm\pi$ radians. This is of little consequence in practice since the field of view can be limited to slightly less than 180° to avoid the singularity.

3.2 Geometric Image Correction

Geometric image correction is the process by which the images of one or multiple projectors are registered together on a display surface so as to compensate for the surface shape and provide the viewer with an image of a scene that is free of distortion. In general, the geometric correction that must be performed is a function of the shape of the display surface, the calibration of all projectors, and the location of the viewer. As such, all of this information must be known in a common coordinate system for geometric correction to be successful.

Geometric image correction is fundamentally a ray-tracing problem whose solution is conceptually simple. Consider a scene consisting of simple geometric shapes, as illustrated in Figure 3.3, where the known display surface geometry, viewing location, and projector calibration are superimposed. Now consider a pixel x in the image space of the projector. As shown in the figure, this pixel is projected along a ray that intersects the display surface at some point A. The exact position of A depends on the shape of the display surface and the projector calibration. The appropriate color for the pixel x is then determined by the color of the point B, which is the intersection of the scene with the ray originating at the viewing location that passes through the point A. The appropriate color for all other projector pixels can be determined in a similar fashion.

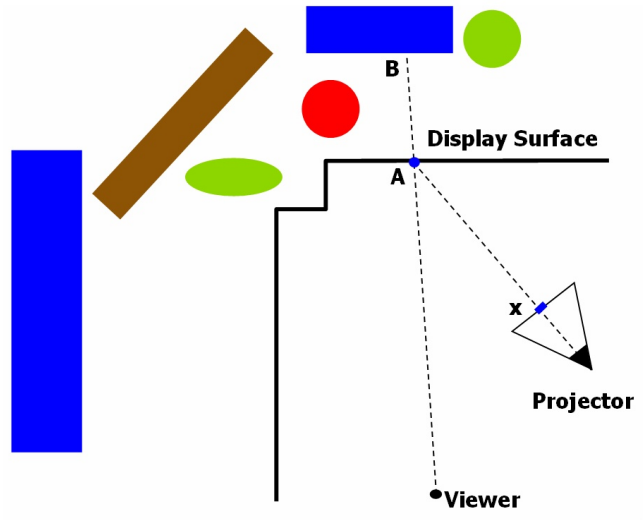


Figure 3.3: Geometric image correction for complex display surface geometry.

3.2.1 Two-Pass Rendering

Though ray-tracing is quite computationally intensive, two-pass rendering [Raskar et al., 1998] is a simple, but powerful technique developed to perform geometric correction efficiently on graphics hardware. The two-pass rendering algorithm assumes that the shape of the display surface has been measured (and is available as a polygonal mesh) and that all projectors in the display have been calibrated so that a projection matrix is available for each. The location of the viewer is also assumed to be known in the coordinate system of the display surface and the projector calibration.

Two-pass rendering is performed independently for each projector and proceeds as follows. In the first pass, as illustrated in Figure 3.4, the scene is rendered from the viewer's perspective with a frustum that overlaps the image extent of the projector on the display surface. This image rendered in pass one is the desired image that the user is to observe on the surface. In the context of the previous ray-tracing example, the desired image is a discrete sampling of the rays originating at the viewpoint that pass through the surface and intersect the scene.

In the second pass, the desired image is warped into the perspective of the projector. This process compensates for the surface shape, resulting in the viewer observing the desired image on the surface. The warping process is illustrated in Figure 3.5. Using a technique called projective texturing, which is supported on modern graphics hardware, the desired image rendered in pass one is mathematically

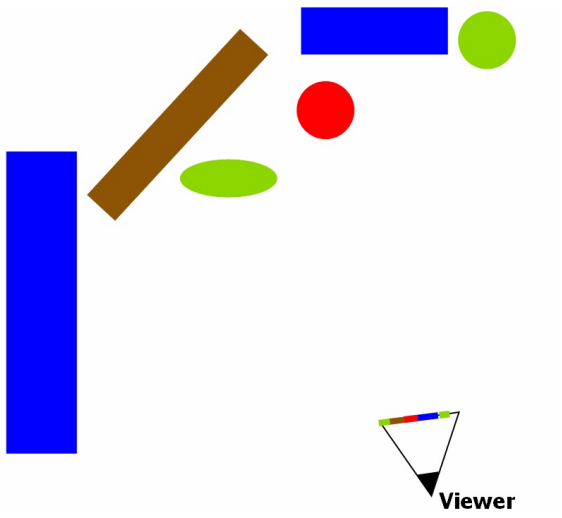


Figure 3.4: The first pass of two-pass rendering.

projected onto the vertices of the display surface model. The textured model is then rendered from the perspective of the projector using its projection matrix to produce the image that the projector must display in order to compensate for the shape of the surface. In the context of the ray-tracing example, this process is, in effect, remapping rays between the viewer and the projector.

The two-pass rendering algorithm, as presented here, implicitly assumes a pinhole perspective model since the projective texturing step in pass two models the projector as a linear device using a projection matrix. In Chapter 4, a method of extending the two-pass rendering algorithm to also include a distortion model for the projector lens will be described.

3.3 Two-View Geometry

The geometry of two views is the minimum number of views required to fully constrain the 3D geometry of a scene and is thus of particular interest in Chapter 5 in formulating a distributed cooperative framework for continuous calibration in multi-projector displays. Much of two-view geometry is centered around epipolar geometry, which stipulates that given a point x_l in one view, its corresponding point x_r in the second view must lie along a line called an *epipolar line*. This stems from the fact that there exists a plane, called the *epipolar plane*, that spans x_l, the two camera centers C_l and C_r and the

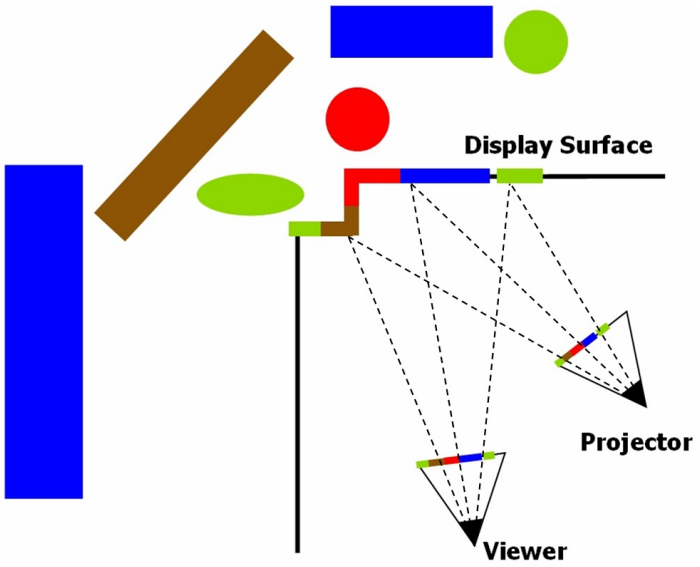


Figure 3.5: The second pass of two-pass rendering.

actual 3D scene point. As illustrated in Figure 3.6, the epipolar plane projects to a line l_r in the second view on which the point x_r must lie. The actual location of x_r depends on the depth of the scene.

A convenient algebraic entity commonly used to represent the epipolar geometry of two views is the *fundamental matrix*, typically denoted as F. Given any image correspondence between two views $x_l \leftrightarrow x_r$, such that x_l and x_r arise from the same 3D point in the scene, the fundamental matrix is a 3×3 matrix that satisfies the homogeneous equation $x_l F x_r = 0$. Using the fundamental matrix, it is simple to compute the epipolar line l_r in the second view that corresponds to the point x_l in the first via $F^T x_l = l_r$ or in the opposite direction $F x_r = l_l$.

The fundamental matrix can be efficiently computed from image correspondences using the normalized 8-point algorithm [Longuet-Higgins, 1981], which requires at least 8 image correspondences in general (non-degenerate) position. Once it is known, it can be used for a variety of applications, such as validating the correctness of image correspondences or directing the search for further correspondences.

An algebraic entity that can be used to represent the special case of two-view planar scene geometry is the *homography*, which maps a point in one view to its corresponding location in the second view without having to search along an epipolar line. Given any correspondence between two views $x_l \leftrightarrow x_r$, the homography H between the two views *induced by a plane* satisfies the equation $H x_l = x_r$.

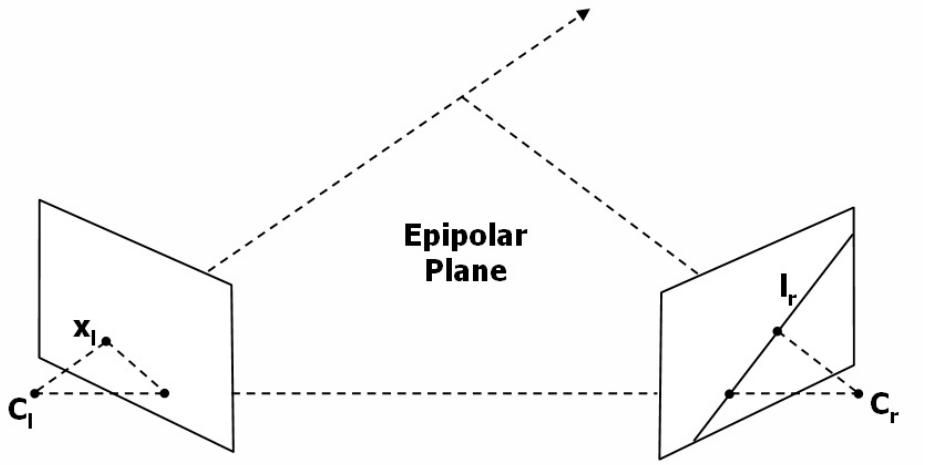


Figure 3.6: The epipolar geometry of two views.

A homography can be computed from a set of 4 image correspondences between two views using DLT [Hartley and Zisserman, 2003]. As with the fundamental matrix, a homography can be used to validate the correctness of image correspondences and direct the search for further correspondences.

3.4 Estimation

Parameter estimation is a branch of mathematics that deals with the estimation of parameters using measured or empirical data. One tool for parameter estimation, the Kalman filter, will prove most useful in Chapter 5 for processing camera image correspondences. In practice, one must also deal with possibility of *false* image correspondences, a measurement in one camera view that is incorrectly identified as corresponding to the same scene point as a measurement in a second camera view. One way of detecting these false correspondences is by applying the RANSAC algorithm to an estimation problem. A brief introduction to RANSAC is thus provided in Section 3.4.2.

3.4.1 The Kalman Filter

The Kalman filter [Kalman, 1960] is a set of mathematical equations used to estimate the state of a random process (a system whose state evolves over time according to some probability distribution) using noisy measurements of its output. While a brief introduction to the filter is provided here, an excellent, and much more thorough, practical introduction can be found in [Welch and Bishop, 2006].

The Kalman filter has proved useful in many areas of computer vision including parameter estimation and tracking, where its recursive nature (the previous state estimate is used as input in estimating the current state) lends itself well to numerical computation. In its standard form, as discussed in this section, the Kalman filter has been shown to be optimal for the problem of discrete-time linear filtering. There are several variations on the standard filter, and one of these in particular, the *extended* Kalman filter, will prove useful in this work and will be subsequently discussed.

The Kalman filter is a stochastic estimator that maintains estimates of both the mean of the process state distribution (state estimate) and the state estimate error covariance (measure of uncertainty in the state estimate). In its standard form, the Kalman filter assumes the process whose state $x \in \Re^n$ is to be estimated is linear and evolves over time according to a discrete-time stochastic difference equation of the form

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}. \tag{3.7}$$

In the above equation, called the *process model*, A is an $n \times n$ matrix, with n the length of the state vector x, that relates the state at the previous time step $k-1$ to the state at the current time step k. The vector $u \in \Re^l$ is optional control input that is related to the state by the $n \times l$ matrix B. Finally, w is a random variable that models random variations in the process over time.

As an example of a process that follows this model, consider a remote-controlled robot whose state consists of its position and velocity. Ignoring any controls or randomness, the position of the robot will have changed between time steps k and $k-1$ by an amount equal to its present velocity. Thus, the action of the matrix A should be to sum the robot's position and velocity at time $k-1$ to obtain its position at time k, leaving its velocity unchanged. Any controls applied to the robot by its remote-controller during this time would be represented in the vector u, with the matrix B describing how these controls translate into a change in position of the robot. Finally, some noise is added to represent the fact that at any point in time, it is not possible to know with complete certainty the true position and velocity of the robot.

The *measurement model* of the filter describes how measurements of the process output are related to its state. In the standard Kalman filter, this relationship is assumed to be linear with measurements

z_k at time k related to the state through an $n \times n$ matrix H as follows:

$$z_k = Hx_k + v_k, \tag{3.8}$$

where v is a random variable used to model noise in the measurements.

Given a process and measurement model, the Kalman filter acts as a predictor-corrector, where at each time step a prediction of the current process state is produced that is later corrected using measurements of the process output. This predictor-corrector paradigm is embodied in the filter through its two phases of operation, the so-called *time* and *measurement update* phases.

The time update phase (prediction) is responsible for propagating the filter state and error covariance forward in time from the previous step, to produce *a priori* state $\hat{x}_k^-$ and error covariance P_k^- estimates at time k. The standard set of filter update equations used in this phase come directly from the process model and are as follows

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3.9}$$

$$P_k^- = AP_{k-1}A^T + Q, \tag{3.10}$$

where Q is the called the *process noise covariance* and is the covariance of the random variable w from Equation 3.7. The purpose of Q is to add additional uncertainty to the error covariance due to random variations in the process that arise *between* filter updates.

In the measurement update phase (correction), the measurements at time k are used in conjunction with a set of measurement predictions to correct the a priori state and error covariance estimates into *a posteriori* state and error covariance estimates. The standard filter update equations for this phase are as follows

$$K_k = P_k^- H^T \left(H P_k^- H^T + R \right)^{-1} \tag{3.11}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H \hat{x}_k^- \right) \tag{3.12}$$

$$P_k = \left(I - K_k H \right) P_k^- . \tag{3.13}$$

In these equations R is the called the *measurement noise covariance* and is the covariance of the random variable v from Equation 3.8. The matrix K, called the *Kalman gain*, is used to weight a residual term that is the difference between the actual measurements collected at time k, z_k, and a prediction of what the measurements should have been according to the measurement model and the current *a priori* state estimate $H\hat{x}_k^-$. The above equation for K is called the *optimal gain* since it results in *a posteriori* state and error covariance estimates such that the *a posteriori* error covariance (uncertainty) is minimized.

The effect of the Kalman gain matrix is heavily dependent on the amount of process noise Q relative to the amount of measurement noise R. As the measurement noise decreases, the gain K gives more weight to the residual as opposed to the *a priori* state estimate, in effect placing more trust in the actual measurements. In the same vein, as the *a priori* state error covariance P_k^- decreases, the gain K gives less weight to the residual, in effect placing less trust in the actual measurements and more trust in the process model.

The Extended Kalman Filter

Some systems are inherently non-linear or have a non-linear relationship between measurements and state. While the standard Kalman filter cannot be applied to such systems, it is possible to apply a variation called the *extended Kalman filter*. The extended Kalman filter works by linearizing the process and measurement functions around the current state and error covariance estimates.

In the extended Kalman filter, the process is assumed to be governed by the equation

$$x_k = f\left(x_{k-1}, u_{k-1}, w_{k-1} \right), \tag{3.14}$$

where f may be non-linear in its parameters. Similarly, the relationship between measurements and the process state is represented by the non-linear function

$$z_k = h(x_k, v_k). \tag{3.15}$$

The operation of the extended Kalman filter is much the same as a standard Kalman filter and is divided into time and measurement update phases (prediction and correction, respectively).

In the time update phase, the following equations are used to propagate the state and error covariance forward in time

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{3.16}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{3.17}$$

where A_k is the Jacobian of f with respect to x evaluated at $(\hat{x}_{k-1}, u_{k-1}, 0)$, and W_k is the Jacobian of f with respect to w evaluated at $(\hat{x}_{k-1}, u_{k-1}, 0)$.

In the measurement update phase, we correct the *a priori* state and error covariance based on the measurements collected at time k. The equations used for the extended Kalman filter are as follows

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R V_k^T\right)^{-1} \tag{3.18}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - h\left(\hat{x}_k^-, 0\right)\right) \tag{3.19}$$

$$P_k = (I - K_k H_k) P_k^-, \tag{3.20}$$

where H_k is the Jacobian of h with respect to x evaluated at $(\hat{x}_{k-1}, 0)$, and represents the sensitivity of the measurements to changes in the state. V_k is the Jacobian of h with respect to v evaluated at $(\hat{x}_{k-1}, 0)$.

While the extended Kalman filter cannot be shown to be optimal in the sense that the *a posteriori* error covariance is minimized, it is applicable to a more general set of processes. As we will see later, it can be applied to the problem of estimating the poses of projectors using camera image measurements.

Observability

In the estimation of parameters, it is possible that the input data that is used may be insufficient to fully constrain a solution. This is seldom an all-or-nothing characterization since some portion of the solution may be constrained, while other parts are not. For example, when estimating the pose of a projector, it is possible that the z component of its pose may be constrained, while its x and y location are not. Of course, it is also possible that some combination of parameters may be unconstrained, for example the location of the projector along the vector $< 0.5, 0.5, 0 >$.

A parameter, or possibly some linear combination of parameters, that is constrained by the input data is termed *observable*, while unconstrained parameters are said to be *unobservable*. In the case of a time-invariant Kalman filter, as has been described in this section, there is a simple test for observability that involves testing the rank of the so called *observability matrix*:

$$O = \begin{bmatrix} H^T & A^T H^T & \left(A^T\right)^2 H^T & \dots & (A^T)^{n-1} H^T \end{bmatrix}, \tag{3.21}$$

where the state is observable if the observability matrix O has row rank n, the dimension of the system state vector [Grewal and Andrews, 2008]. As can be seen from the use of the matrix A in the formulation of the observability matrix, the concept of observability includes the concept of time since repeated observations of the system over time may allow the full state information to be accumulated.

Uncertainty

The Kalman filter provides an estimate of the uncertainty in the state parameters via the error covariance matrix P. Since parameters may be uncertain along any vector in the state space, it is useful to have a general technique for determining what directions in the state space are most uncertain. For example, high amounts of uncertainty may indicate that some parameters, or directions in the state space, are unobservable. Also, uncertainty in the parameters of a dynamic system may ebb and flow over time due to the amount of information about the state parameters the filter is able to learn from the measurements. Knowledge of the most uncertain direction in the state space can then be used to guide measurement collection, a process called *measurement selection*, where measurements are selected specifically for their ability to reduce the amount of uncertainty in the state estimate.

The direction(s) of maximum uncertainty in the state space can be determined by performing a singular value decomposition (SVD) on the error covariance matrix P of the filter. The SVD algorithm will decompose the $n \times n$ matrix P such that $P = UDV^T$, where U and V are $n \times n$ orthogonal matrices, and D is an $n \times n$ diagonal matrix of the singular values of P. The direction of maximum uncertainty is then the Eigenvector of P that corresponds to its maximum singular value, which is simply the first column of U or V since P is a symmetric matrix. The next greatest direction of uncertainty is the second column of U or V and so forth.

Sequential Measurement Processing

As opposed to closed form solutions for parameter estimation, such as the DLT algorithm for computing a homography between two images, the Kalman filter assimilates measurements across time and it is thus possible that *over time* the state may be become observable, even though individual sets of measurements processed by the filter are insufficient to fully constrain a solution.

Taking full advantage of this property, [Welch and Bishop, 1997] showed that a "SCAAT" (single constraint at a time) approach to Kalman filtering, where single measurements or observations are processed sequentially, can have several advantages over processing measurements in batch. In multi-sensor systems, this has the advantage that measurements can be processed immediately as they are available from each sensor rather than attempting to synchronize the sensors and then fuse the measurements into a data set that fully constrains the state estimate. The authors also note that a sequential processing approach can also be advantageous in systems where sensor synchronization and data fusion are not an issue since the computational intensity of performing filter updates is lessened due to the smaller matrices that must be processed. A potential disadvantage of this approach over batch processing, however, is that the filter may lose some tolerance to outlying measurements or measurements that may not adhere to the underlying assumptions made by the filter, leading to a greater potential for instability. It is thus important to randomize the processing of measurements in a sequential approach.

3.4.2 Robust Estimation using RANSAC

RANdom SAmple Consensus (RANSAC) [Fischler and Bowles, 1981] is a data-fitting tool that performs well even for data sets that may contain a large proportion of outliers. It has been successfully applied to many problems from computer vision such as the computation of homographies and the fundamental matrix. The RANSAC algorithm will prove useful later in developing a distributed cooperative framework for continuous calibration in multi-projector displays that is robust against outlying camera image measurements.

The basic concept is to first identify the smallest sample size s for which the model can be fit and to then iteratively fit the model to randomly selected samples of size s until it can be said with a certain probability that at least one chosen sample was free of outliers. For each selected sample, the number of data points that are within a certain tolerance of the fitted model is determined. This is called the sample's *support*. During the iteration, the sample with the largest support is maintained and once the iteration completes, the samples in its support (the *inliers*), are used to fit the final model.

As a practical example, consider the problem of fitting a plane to a set of N points in 3-space. Since any three points determine a plane, we iteratively choose three points from the set at random, compute the plane spanned by the points, and then determine the number of points (the sample's support) that are within the accepted tolerance of the sample plane. Once a sufficient number of iterations have been completed (it can be said with probability p that at least one sample contained no outliers), we fit a least-squares plane to the inliers of the sample of largest support. The results of this algorithm for a real data set are shown in Figure 3.7, where the sample of largest support is shown as a green triangle and its inliers are shown in blue. The red points in the figure are outside the tolerance of the sample plane (outliers).

Given the proportion of inliers and the size of the samples, it is relatively simple to compute the number of iterations required such that, with probability p, at least one sample is free from outliers. If w is the probability that a selected data point is an inlier and s is the size of a sample, then the probability p that after N iterations at least one sample contains no outliers is

$$p = 1 - (1 - w^s)^N. \tag{3.22}$$

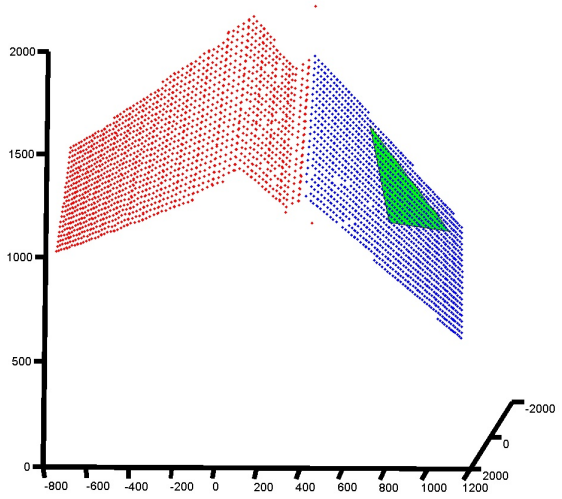


Figure 3.7: Results of fitting a plane to a point cloud using RANSAC. Inlying points are shown in blue. The sample of largest support is shown as a green triangle.

Given a desired probability p, we can then compute the number of iterations N that must be performed via

$$N = log(1-p)/log(1-w^s). \tag{3.23}$$

Unfortunately, in practice the proportion of inliers is not typically known. This information can, however, be estimated during iteration by examining the proportion of inliers for each sample. This leads to the adaptive RANSAC algorithm where, initially, the proportion of inliers is assumed to be small and is updated as iteration continues to adaptively determine the maximum number of iterations N that must be performed. This process is outlined in Algorithm 1.

Input : Data Set S, Tolerance t, Probability p
Output: Fitted model M

1: $N = \infty$
2: $i = 0$
3: **while** $N > i$ **do**
4: Select a sample s from S at random
5: Fit the model to s, yielding M_s
6: Determine the proportion w of data points in S that are within t of M_s
7: Update N via Equation 3.23 using desired probability p.
8: **end while**
9: Fit the model to the inliers of the sample of largest support.

Algorithm 1: ADAPTIVERANSAC

CHAPTER 4

PROJECTIVE DISPLAY WITH CORRECTION FOR DISPLAY SURFACE GEOMETRY AND EXTREME LENS DISTORTION

Projectors equipped with wide-angle lenses (such as a fisheye lens) have an advantage over traditional projectors in creating large display areas even in cases where the projector is positioned close to the display surface (Figure 4.1). In front-projection configurations, this wide-angle capability may allow the projector to be positioned between the viewer and the display surface, which has the added advantage of eliminating the problem of user-shadowing. In practice, however, wide-angle lenses typically exhibit increased or possibly extreme lens distortion that must be corrected to allow accurate geometric image correction.

A number of displays have been designed to take advantage of the capability afforded by wide-angle lenses through the use of specialized display surfaces. The Elumens VisionStation® uses a single wide field-of-view projector and a specialized display screen to provide an immersive experience to a stationary user. Konieczny et al. [Konieczny et al., 2005] use a fisheye lens projector to allow a user to explore volume data by manipulating a tracked sheet of rear-projection material.

While these examples make use of specialized display surfaces, a general solution to the problem of geometric image correction in the case of arbitrary display surface geometry and extreme projector lens distortion is desirable. While the two-pass rendering technique described in Chapter 3 provides a general solution to the problem of geometric image correction for arbitrary display surfaces, it

Figure 4.1: An immersive display built with a single fisheye projector.

assumes a pinhole perspective model for the projectors and is not suited to displays where significant lens distortion is present.

To address this issue, this chapter describes an extension to the two-pass rendering algorithm that provides geometric image correction for multiple projectors and arbitrary display surfaces even under the condition of extreme projector lens distortion. It is also shown that the obvious extension to two-pass rendering, adding an additional (third) lens distortion correction pass, cannot make use of the full field-of-view of the projector without introducing strong aliasing artifacts. Finally, perspectively correct results using a fisheye-lens projector displaying into a room corner are demonstrated.

4.1 Lens Distortion Correction for Cameras

Due to the dual relationship between projectors and cameras, a brief introduction to lens distortion estimation and correction for cameras is provided here.

Lens distortion estimation in cameras is typically performed as an optimization on top of an initial linear estimation (pinhole perspective model with no distortion) that approximates the behavior of the camera. For example, we can estimate the projection matrix of a camera via a set of 3D-2D point correspondences using the Direct Linear Transformation (DLT) technique [Abdel-Aziz and Karara, 1971,

Faugeras and Toscani, 1987]. This projection matrix can then be used as an initial guess in a non-linear optimization that estimates the parameters of the full model to minimize the sum of squared reprojection errors.

If $P = K[R|t]$ is the output of DLT and $X_{1..N}$ is a set of 3D points with a known set of 2D correspondences $x_{1..N}$, the non-linear technique should minimize

$$\sum_{i=1}^{N} dist(K\delta_{in}([R|t]X_i), x_i))^2, \tag{4.1}$$

where δ_{in} is the distortion function of the camera for light *entering* the lens as described in Chapter 3.

When lens distortion correction is performed on camera images, the goal is to take a distorted image captured by the device and use the distortion model to produce the undistorted image that would have been captured by a perfect pinhole camera. This process can be performed in one of two ways. Either we color the undistorted image at each pixel by sampling from the captured distorted image, or we splat each pixel of the captured distorted image onto the pixels of the undistorted image in some way. The first technique requires calculation of δ_{in}, while the second would require calculation of δ_{out}. Since a sampling procedure is typically preferred, lens distortion properties for cameras are usually calibrated in a way that allows the calculation of δ_{in} when the distortion model is not easily inverted,

If δ_{in} is known, a pixel $p = [x, y, 1]^T$ in the desired undistorted image will map to a pixel $p' = K\delta_{in}(K^{-1}p)$ in the captured distorted image. The captured image can then be filtered around p' to produce a color for p in the undistorted image.

4.2 Two-Pass Image Correction for Wide-Angle Lenses

The basic two-pass multi-projector image correction technique described in Chapter 3 works well under the condition that the geometric properties of the display devices do not deviate significantly from the pinhole perspective model. Significant deviations from this model can result in obvious visual artifacts such as ghosting in projector overlap regions and miscorrected imagery. This section describes an extension of Raskar's original two-pass rendering algorithm that is able to overcome these issues by incorporating correction for projector lens distortion and non-pinhole lens models.

4.2.1 Modified Projector Calibration

Lens distortion can be incorporated into the geometric model of a projector by continuing Raskar's original treatment of the projector as the dual of a camera. Chapter 3 described some lens and distortion models commonly used for cameras. These same models can be used for projectors by modifying the calibration process slightly to account for projector/camera duality.

In contrast to cameras, when dealing with projectors we are interested in the light that travels outward from the device, making it of greatest practical value to calibrate for projector lens distortion in a way that allows the calculation of δ_{out}. This models how light is distorted as it *exits* the lens and allows the back-projection of each projector pixel into a ray in world space in a manner that accounts for distortions in non-pinhole lenses.

Following this observation, the process used to calibrate projectors in the face of lens distortion is identical to that described in Section 4.1 for calibrating cameras with lens distortion, except that we replace the minimization function for cameras in Equation 4.1 with

$$\sum_{i=1}^{N} dist(PX_i, K\delta_{out}(K^{-1}x_i))^2. \tag{4.2}$$

Here we have distorted the projected feature locations $x_{1..N}$ at each iteration using δ_{out}, which can model either a pinhole lens with radial and tangential distortion characteristics or a non-pinhole lens model. For models such as the Brown model that are not easily invertible, this will yield distortion coefficients allowing us to pre-distort an image before projection so that the projector becomes a linear device.

4.2.2 Modified Geometric Correction

The previous section described how a projector can be calibrated in a way that allows compensation for lens distortion by pre-distorting an image before projection. Given such a calibration, it would be a simple process to add an additional third pass to the two-pass correction algorithm described previously to perform this operation. To determine the color for a pixel p in the desired compensation image, we can filter the image rendered during pass two around the pixel $p' = K\delta_{out}(K^{-1}p)$.

The problem with this technique when using wide-angle lenses is that rendering in pass two will

not generate enough imagery to fill the field-of-view of the projector *after* distortion correction. This occurs because the pass two image will be rendered with the pinhole perspective model using the intrinsics of the wide-angle lens. This greatly reduces the field-of-view since a wide-angle lens has a much greater field-of-view than a pinhole lens of the same focal length. One solution to this problem would then be to calculate the set of intrinsics that a pinhole lens would require to have a field-of-view comparable to the projector's wide-angle lens.

Unfortunately, for extremely wide-angle lenses, such as fisheye lenses, this technique has the downside of introducing aliasing artifacts. The reason for this is illustrated in Figure 4.2, which was created using actual data from an experimental f-theta fisheye lens projector donated by D'nardo Colucci of The Elumenati, LLC. The rectangle situated at the origin represents the border of an image given to the projector on the image plane before it is distorted by the lens. This is the same as the region of the image plane that the K matrix of the projector will transform to valid pixel coordinates in the projector image. The contour surrounding the image depicts the extent to which the borders of the image are distorted by the lens of an experimental f-theta fisheye lens projector when the field-of-view is limited to 178° in order to avoid the singularity in the model at 180°.

If 178° of the horizontal projector field-of-view is to be filled after distortion correction, the new intrinsics K' must transform a region enclosing the entire distorted contour into valid pixel coordinates. Since K is an upper-triangular matrix, the region of valid pixel coordinates must form a parallelogram on the image plane. Clearly, if such a region is to enclose the distorted border, the pixels of the pass-two texture must be stretched over a much larger spatial extent, leading to a low-pass filtering of the pass-one texture and aliasing during distortion correction due to large changes in pixel density over the extent of the distorted region. Also, those pixels not falling within the convex hull of the distorted contour are effectively wasted since no pixels in the compensated image will map to their location.

While increasing the resolution of the textures rendered during passes one and two of the pipeline can reduce the amount of aliasing, it remains significant up to the maximum texture size that current hardware is able to render. Figure 4.3a depicts the aliasing that results when this approach is used to generate a 178° field-of-view image using the fisheye projector. Figure 4.3b shows that significant aliasing is still present when rendering in passes one and two is performed at four times projector resolution.

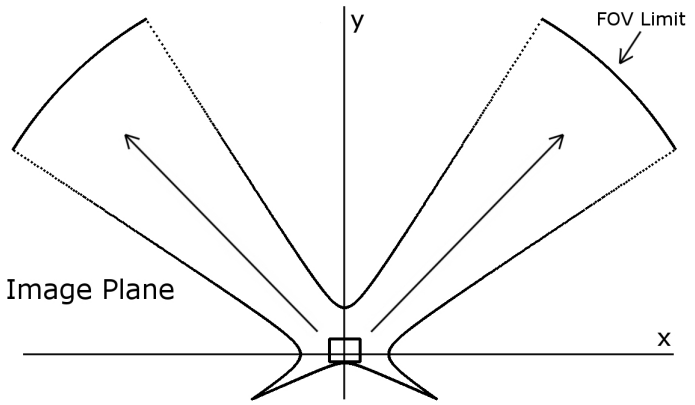


Figure 4.2: Distortion resulting from fisheye projector lens. Compare this to the pincushion lens distortion of Figure 3.1

A Better Approach

This section describes a novel two-pass rendering solution that eliminates the aliasing concerns that a three-pass technique can introduce. The objective is to be able to simulate a non-pinhole rendering model in pass two, eliminating the need for a third pass lens distortion correction step.

In pre-process, the projector calibration, including distortion properties, is used in conjunction with the display surface model to determine the 3D location on the display surface that each projector pixel illuminates. Given the projector calibration $P = K[R|t]$ and δ_{out}, each pixel $x_i = [u_i, v_i, 1]^T$ of the projector is back-projected to produce a ray

$$r(\alpha) = C + \alpha R^{-1} \delta_{out}(K^{-1} x_i). \tag{4.3}$$

This ray is then intersected with the polygons of the display surface model yielding a 3D point on the display surface. This process is repeated until the mapping has been performed at the resolution of the projector.

Using this 2D-3D mapping, it is possible to correct for both the display surface geometry and lens distortion in the second pass by projecting each projector pixel's 3D location into the pass-one texture to produce the pixel's output color. If both the projector and display surface remain static during display, this 2D-3D mapping will remain fixed even though the position of a head-tracked viewer may change.

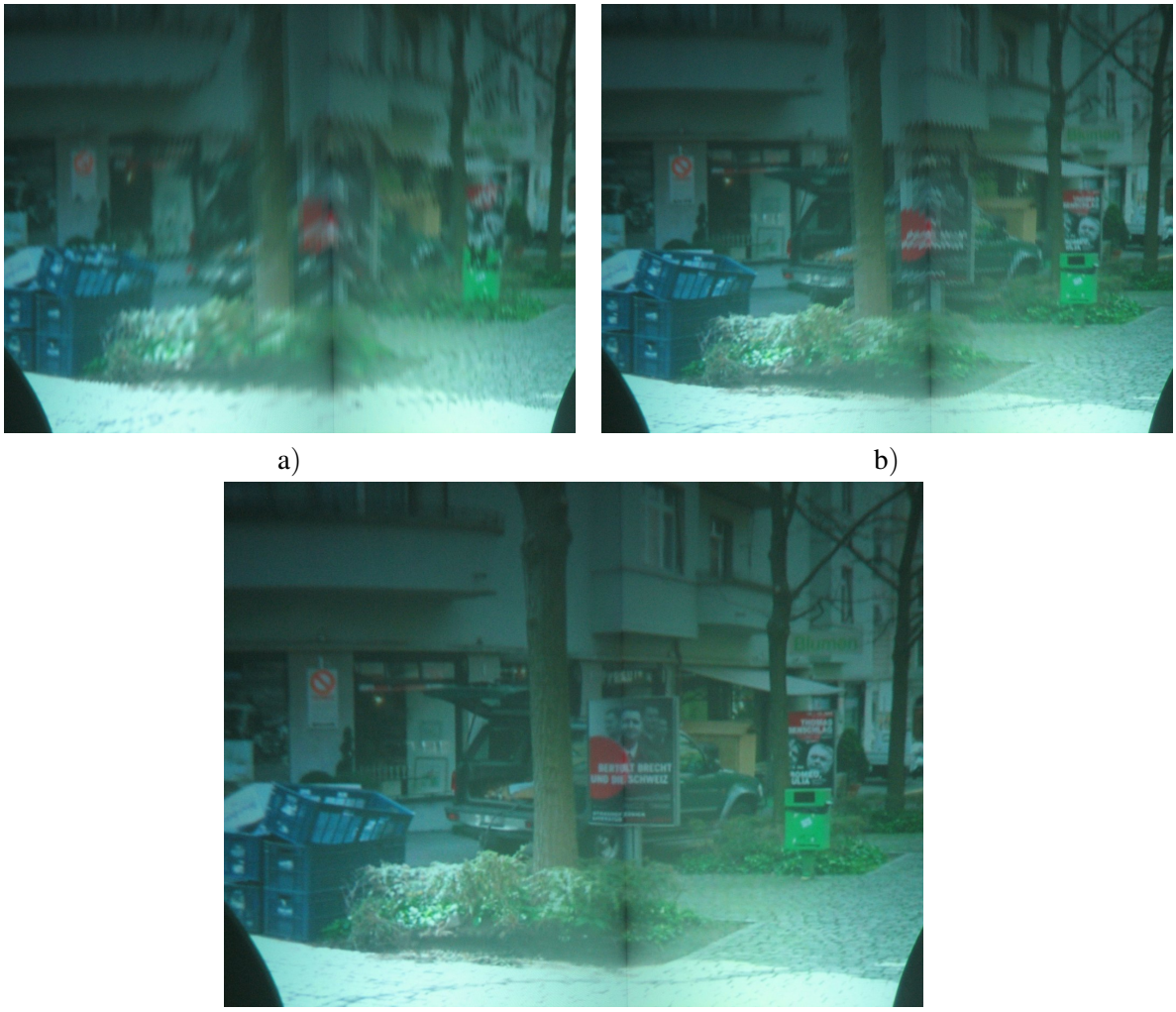


Figure 4.3: Three increasingly effective methods for performing geometric compensation in the case of a fisheye projector lens. a) Three-pass correction for display surface geometry and fisheye distortion. b) Three-pass correction super-sampled fourfold. c) Two-pass rendering extended to incorporate fisheye lens model without super-sampling.

Graphics card vendors now produce consumer cards with floating-point pipelines that allow the use of textures consisting of four 32-bit floating-point values per pixel. This technology is used to store the projector's 2D-3D mapping directly on the graphics card. A floating-point texture is created at the resolution of the projector where the floating-point location (x,y,z) on the display surface that a pixel illuminates is stored as its (r,g,b) elements in the texture. The alpha component of each pixel in the texture can also conditionally be used as a flag to indicate that the pixel should be left black. This is useful when the geometry of the display surface model does not fill the entire field-of-view of the projector.

At render time, a pixel shader takes as input the floating-point texture of display surface geometry, the desired image from pass one, and the viewing matrix of the viewer modified to act as a texture matrix. The shader simply looks up the vertex information for the pixel it is currently shading and projects the vertex into the pass-one texture using the texture matrix to produce an output color. This GPU implementation allows correction to take place at interactive frame rates.

4.2.3 Modified Edge Blending to Account for Lens Distortion

In addition to the two-pass algorithm to correct for image distortions due to arbitrary display surfaces, [Raskar et al., 1999b] also describes a simple technique for performing edge blending in multi-projector displays that eliminates areas of increased brightness where projectors overlap. The basic idea is to compute an alpha mask for each projector that gives the attenuation value to apply at each pixel in order to blend smoothly between overlapping projectors.

The alpha masks are computed by observing projector overlap regions with a camera. Attenuation values are then computed in the image space of the camera for each projector by taking into account the number of projectors overlapping at each camera pixel and the distance to the convex hull of each projector's contribution in the camera image. The attenuation value for projector m at camera pixel (u,v) is computed as

$$A_m(u,v) = \frac{\alpha_m(m,u,v)}{\sum_i \alpha_i(m,u,v)}. \tag{4.4}$$

In the above equation, $\alpha_i(m,u,v) = w_i(m,u,v) \cdot d_i(m,u,v)$ where $w_i(m,u,v)$ is 1 if (u,v) is within

the convex hull of projector i and 0 otherwise. The $d_i(m,u,v)$ term is the distance from camera pixel (u,v) to the convex hull of projector i in the camera image. This technique produces weights that sum to one at each camera pixel and decay gradually to zero at projector edges. Since weights are computed in camera image space, each projector's weights are transformed into its own image space using the two-pass image correction algorithm.

Since the generation of the alpha masks for each projector relies on the use of the two-pass algorithm, the original technique cannot be used to generate accurate alpha masks for projectors that do not fit the pinhole model. A slightly different approach to the problem is to use the geometric information from calibration, including the lens distortion model of the projector, to produce an alpha mask for each projector without the additional use of a camera required by the original technique. Alpha masks for each projector can be calculated in this way as outlined in Algorithm 2.

Input : Display surface model and calibration for each projector
Output: A projector-resolution attenuation mask A_i for each projector

for each projector i **do**
  for each pixel j of projector i **do**
    $r \leftarrow$ back-project j using Equation 4.3
    $X \leftarrow$ intersect r and display surface
    $sum \leftarrow 0$
    for each projector $k \neq i$ **do**
      $x \leftarrow$ project X using Equation 3.4
      if x within displayable area **then**
        $sum \leftarrow sum +$ min dist to projector k's image border at x
      end if
    end for
    $m \leftarrow$ min dist to projector i's image border at j
    $A_i[j] = \frac{sum}{sum+m}$
  end for
end for

Algorithm 2: GENALPHAMASKS

4.3 Results

A number of different display configurations have been explored using the calibration and image correction approach described in this chapter for projectors with lenses deviating from the pinhole model. Each display was calibrated by observing projected structured light patterns with a stereo

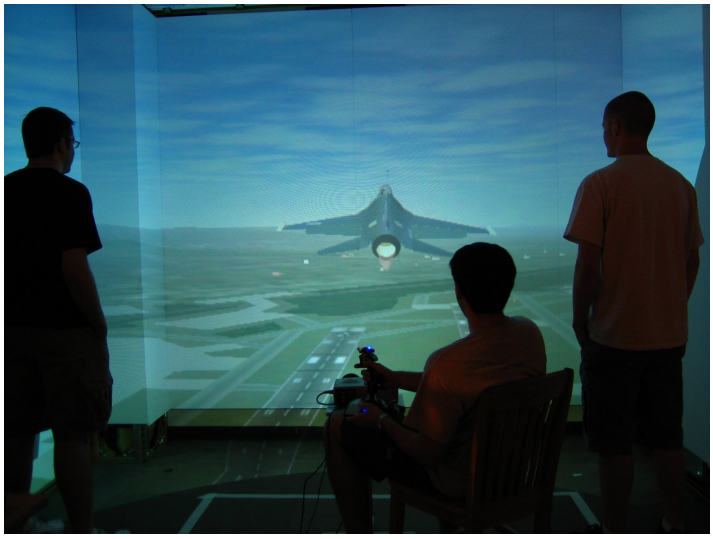


Figure 4.4: Fisheye projector used in a flight simulator application.

camera pair used to reconstruct the display surface via triangulation. A plane-extraction technique was used [Quirk et al., 2006] to extract a polygonal display surface model from the reconstructed point cloud that closely approximated the piece-wise planar display surfaces in the experimental display.

Figure 4.4 depicts a fisheye projector being used in a virtual reality application to illuminate a multi-wall surface. An overhead view of this configuration is provided in Figure 4.6 left, which clearly shows the close proximity of the projector to the display surface. The panorama of Figure 4.9 shows the view from a user's position very near the projector. This illustrates the nearly 180° immersive environment created by the fisheye projector. Compare this with Figures 4.6 right and 4.7 where a conventional projector was used, showing the difference in the size of the imagery that is produced.

The panorama of Figure 4.8 shows a fisheye projector displaying on this same multi-wall surface without any consideration for lens distortion. Note how the imagery is warped by the lens, prohibiting proper correction for the distortion due to the display surface geometry. Contrast this with the quality of the correction that is achieved in Figure 4.9 using the methods described in this chapter. Any apparent distortion due to the lens has been removed and distortions due to the display surface geometry are also well corrected. The correction does have a slight flaw in the right side of the image where there is a 1-2 pixel error in the correction for the corner geometry. This is likely due to slight optical deviations of the fisheye lens from its f-theta model, which could be accounted for in future work.

Figure 4.3c is a close-up of the correction method, which shows that it is able to correct for the image distortion introduced by both the display surface geometry and the fisheye lens without introducing the aliasing artifacts present in Figures 4.3a and 4.3b.

As an illustration of the general applicability of the method, the fisheye projector was combined with a conventional projector to form a multi-projector display. The conventional projector was calibrated using the Brown distortion model while the fisheye projector was calibrated with the f-theta model. The resulting display is depicted in Figure 4.5. This display uses the modified edge blending algorithm described in this chapter. Unfortunately, the algorithm currently does not take into account differences in pixel density or black and white levels between the two projector models, resulting in some edges being only softened.

Figure 4.5: Display system combining a conventional projector and a fisheye-lens projector. Note the accurate geometric registration in the overlap region (center of figure).

4.4 Summary

This chapter has demonstrated how a single fisheye-lens projector can be used in conjunction with camera-based calibration to create a personal immersive display system in an ordinary room without the need for a specialized display screen. This allows images several times larger than that of a conventional projector to be generated at the same distance from the display surface, making it possible

for viewers to stand close to the display surface without shadowing projected imagery.

The distortion introduced by the projector lens is corrected by extending the two-pass rendering algorithm for multi-projector displays that provides perspectively correct imagery to a tracked viewer. This extended method is able to incorporate both conventional projectors with slight lens distortion characteristics and non-conventional fisheye-lens projectors with extreme distortion into a single display without sacrificing support for dynamic viewer tracking. Using programmable commodity graphics cards, this technique is able to take advantage of the extremely large field-of-view afforded by fisheye lenses without introducing undesired aliasing artifacts that can occur when performing lens distortion correction.

Even though this correction algorithm allows fisheye-lens projectors to be used in multi-projector displays without introducing aliasing, there are other sampling issues that should be addressed. Since a projection matrix is used to texture the desired image onto the display surface model in pass two, as the viewer approaches the display surface, the field-of-view of the frustum that must be used to texture the entire display surface geometry may approach 180°. This can lead to substantial aliasing artifacts in the corrected image due to over- and/or undersampling of the pass one image. One solution would be to render multiple views from the viewing position in different directions during pass one. Also, when resampling the desired image during pass two, it is possible to extend the method to take into account the complex ways in which projector pixels may overlap with pixels of the desired image after they are projected onto the display surface model. Currently, the basic bilinear filtering supported by the graphics hardware is used, but ideally the desired image would be filtered using an area-weighted sampling technique that is not limited to four pixels.

A more general lens model such as that described in [Kannala and Brandt, 2006], which allows both pinhole and fisheye lenses to be modeled in the same way, could also be of benefit. This would allow deviations of a lens from its associated lens model to be accounted for, while making it possible to utilize a field-of-view larger than 180°.

Although fisheye-lens projectors can be used to create immersive displays at close proximity to a display surface, they can suffer from loss of brightness near the image periphery. Also, conventional projector lenses may be better suited when the spatial resolution of projected imagery is favored over its size since a fisheye lens will distribute the resolution of the device over a much larger field-of-view.

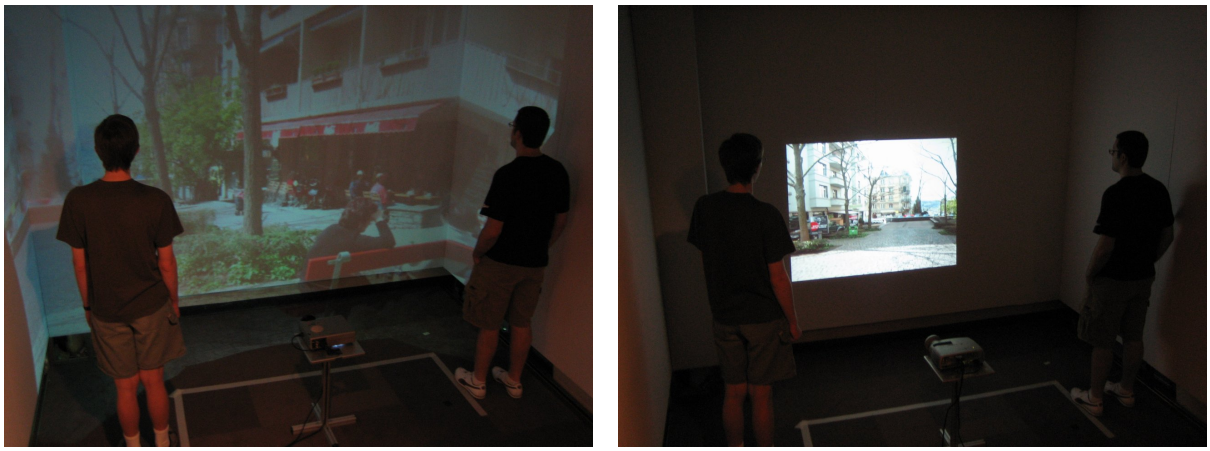


Figure 4.6: Imagery produced by a single fisheye-lens projector (left) versus that produced by a single conventional projector placed at the same location (right).

Figure 4.7: Imagery produced by a conventional projector placed at the same location as the fisheye projector in Figures 4.8 and 4.9.

Figure 4.8: Distorted imagery produced by a fisheye lens projector when modeled as a pinhole lens.

Figure 4.9: Imagery produced by a fisheye-lens projector using extended two-pass rendering.

CHAPTER 5

A DISTRIBUTED COOPERATIVE FRAMEWORK FOR CONTINUOUS MULTI-PROJECTOR POSE ESTIMATION

Projection-based displays have long been used in the creation of large, immersive environments for virtual reality (VR), simulation, and training. These displays have become increasingly useful due to advancements in automatic calibration that allow the images of multiple projectors to be registered together accurately on complex display surfaces, while simultaneously compensating for the surface shape. While these techniques have been shown to be accurate and robust, the geometric aspects of the calibration are typically only computed *prior* to display use. However, even for "fixed" configurations of projectors, vibration of the mounting structure and gravity may cause changes in projector pose over time, ultimately decreasing the quality of the display. In areas where the imagery of multiple projectors overlaps, even a few pixels of error can make certain imagery, such as text, almost useless.

The goal of this work is a framework that provides for continuous and automatic adjustment to even the slightest change in projector poses during display use. This continuous pose estimation should increase the robustness of projection-based displays, while also offering new flexibility. For example, the most suitable positioning of projectors may vary between applications due to certain field-of-view or spatial resolution requirements. In these situations it might be desirable to deliberately reposition the projectors without having to interrupt display use to perform a recalibration of the entire system. It may also be useful in rapid set-up applications by allowing the use of a quick, initial

calibration of the display that is subsequently refined using continuous calibration.

This chapter presents a novel continuous calibration framework for estimating the poses of multiple projectors during actual display use. This framework is designed around "intelligent" projector units (IPUs): a projector augmented with two rigidly-mounted cameras, and paired with a dedicated computer (see Figure 5.1). The IPUs operate in a distributed fashion, each cooperating with its neighbors through the exchange of pose estimates, error covariances, and measurements to continuously estimate all of the poses. In cases where the projection surface is static, this system is able to continuously refine all of the poses, even when IPUs move simultaneously. A simple extension that allows simultaneous estimation of projector pose and information about the display surface, such as its pose or shape, is described in the next chapter.

5.1 Design and Goals

This section describes the design of the distributed cooperative framework presented in this chapter and how the goals of this design are supported by the use of "intelligent" projector units.

5.1.1 Continuous Projector Calibration

The primary goal of this framework is to support rapid set-up and reconfiguration of multi-projector displays. Towards achieving this goal, the framework is designed to continuously estimate the poses of all projectors during actual display use.

In order to estimate changes in projector pose over time, the use of an auxiliary measurement device is required. Cameras were chosen for this purpose due to the inherent duality between cameras and projectors—both use lenses to direct light, but one for the purpose of sensing and the other for the purpose of display. The utility of such combinations of projectors and cameras is described in a variety of previous work [Bhasker et al., 2006, Raskar et al., 2003, Cotting et al., 2004, Zhou et al., 2008, Underkoffler and Ishii, 1998].

This system is designed to be flexible enough to recover the poses of all projectors even in the case that all projectors have been moved. This is important because, even in displays where projectors are not intentionally moved, the poses of all projectors are likely to change slightly over time. Previous

work [Zhou et al., 2008, Cotting et al., 2005] solves this problem only partially by using projector-camera pairs with known calibration as references in re-estimating the poses of other projector-camera pairs that have been moved. This approach fails when *all* projector-camera pairs have been moved, and it is not clear how errors may accumulate over time as devices are moved, recalibrated, and then used to recalibrate other devices.

In order to achieve the goal of allowing any and all projectors to be moved simultaneously, the known display surface geometry is used as an additional constraint in estimating projector pose. This allows the distinction between calibrated and uncalibrated projectors to be eliminated and instead refines the calibration of all projectors continuously over time. In this chapter, we will assume that the geometry of the entire display surface is known a priori and does not change, but will relax this constraint in Chapter 6.

A secondary goal of the current implementation is to allow continuous projector pose estimation to take place without affecting the imagery projected by the user. While techniques for embedding imperceptible patterns [Cotting et al., 2004, Cotting et al., 2005, Grundhöfer et al., 2007] into projected imagery ensure a steady supply of features that can be used as camera image measurements, these techniques are currently limited in the types of projectors that can be used, i.e. DLP® or stereo, and the embedding process requires that some amount of image quality be sacrificed. For this reason, the projected imagery itself is used as a source of camera image features that can be used to estimate projector pose during display use.

5.1.2 Distributed Cooperative Operation

In order to be useful in large displays with many projectors, continuous calibration approaches must be scalable. As described in [Bhasker et al., 2006] a distributed calibration methodology has far greater potential for scalability and fault tolerance than more traditional centralized approaches where all calibration data is aggregated and processed on a single machine.

The design of this distributed cooperative framework allows for scalability by pairing computation with each projector to create a number of self-contained, intelligent units that act in a cooperative fashion, leveraging their combined computational power to estimate the pose of each projector through the exchange of pose estimates, error covariances, and measurements over a local network.

5.1.3 Intelligent Projector Units (IPUs)

Intelligent projector units (IPUs) are the basic building blocks of the system. An IPU, as seen in Figure 5.1, consists of a projector augmented with rigidly-mounted cameras and computation that includes network capability. While the computation component currently consists of a separate PC, future designs will fully integrate the computation with the rest of the unit.

Figure 5.1: An intelligent projector unit (IPU).

5.2 Distributed Computational Framework

This section describes the distributed computational framework for continuous calibration of projector pose in multi-projector displays.

5.2.1 Assumptions

1. The internal calibration of each IPU is fixed and known. (The internal calibration consists of the intrinsic parameters of the projector and both cameras, such as focal lengths, principal points, and lens distortion coefficients, as well as the relative positions and orientations of all three devices.)

2. The geometry of the display surface is static and known (this is relaxed in Chapter 6).

3. Projectors remain mostly stationary, however they may drift over time or occasionally be moved, purposefully or inadvertently, by the user.

5.2.2 General Approach

In the distributed computational framework that is developed here, each IPU is tasked with the responsibility of estimating its own pose and does so by interacting with its peers in a decentralized, peer-to-peer-based fashion. In general, the pose of a camera or projector has six degrees of freedom that correspond to its position and orientation. The three position parameters x, y, z represent the device's center-of-projection, while the three rotational parameters r_x, r_y, r_z represent the orientation of its principal axis.

When the internal calibration of an IPU is known (Assumption 1 in Section 5.2.1) knowledge of the pose of any *one* of the optical devices (projector or either camera) that are part of the IPU is sufficient to completely constrain the poses of all three devices. Taking advantage of this property, one of the two cameras in each IPU is chosen arbitrarily to serve as its "primary" camera, whose pose will be continuously estimated. The pose of an IPU is thus defined to be equivalent to the pose of its primary camera. The pose estimate of the IPU's primary camera can then be transformed into a pose estimate for its projector, for use in warping the projected imagery to register it to the other projectors and compensate for the shape of the display surface.

In this framework, each IPU estimates the pose of its primary camera using image (feature) correspondences between cameras. An image correspondence between two cameras consists of a pixel location in the image of each device that correspond to the same 3D point in the scene. In general, given a measurement in the image of one camera, its correspondence in the image of another camera is determined by the intrinsic and extrinsic calibration of both devices and the geometry of the scene that is observed (the display surface in the case of a multi-projector display).

Local and Remote Correspondences

In continuously estimating the pose of its primary camera, each IPU makes use of two types of image correspondences. The first type consists of correspondences between its primary and secondary

cameras. These will be referred to as "local" correspondences since each IPU can obtain these correspondences independently of the other IPUs. The second type of correspondence that is used in the system are correspondences between an IPU's primary camera and the primary cameras of other remote IPUs. These are referred to as "remote" correspondences.

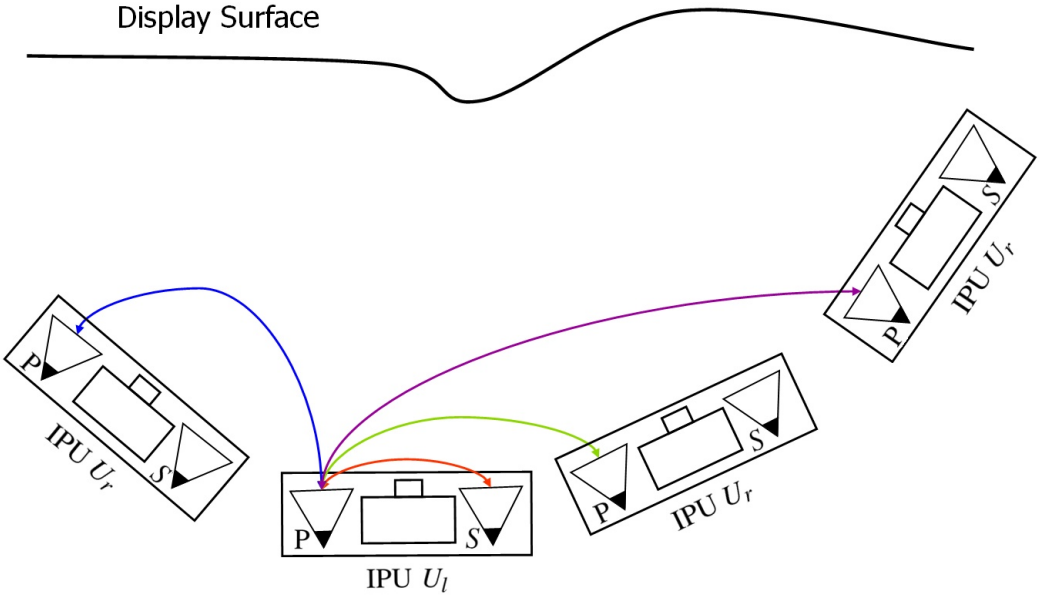


Figure 5.2: The IPU U_l considers itself to be the local IPU. It collects both local correspondences (between its primary and secondary cameras) and remote correspondences (between its primary camera and the primary cameras of remote IPUs).

The concept of local and remote correspondences is illustrated in Figure 5.2 from the perspective of one IPU in a four IPU display. The IPU U_l considers itself to be the local IPU, and the other IPUs are considered to be remote IPUs. The local IPU collects local correspondences between its primary and secondary cameras, as well as remote correspondences between its primary camera and the primary cameras of other remote IPUs.

Both local and remote correspondences produce constraints on an IPU's pose. As seen in Figure 5.3, local correspondences constrain the structure of the display surface that is currently observed by an IPU since its primary and secondary cameras are calibrated as a stereo camera pair. The pose of the IPU is then constrained by requiring that the currently observed surface geometry (shown as blue points in the figure) match the model of the display surface, which is assumed to be static and known (Assumption 2 Section 5.2.1).

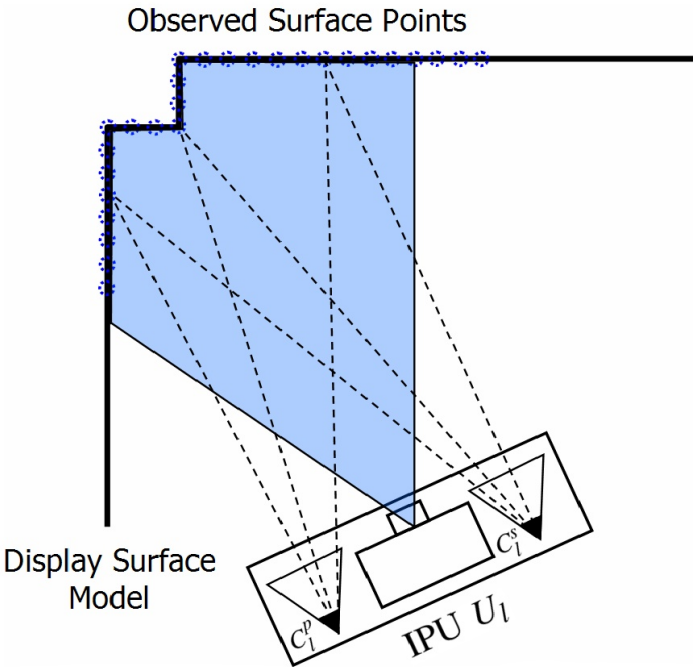


Figure 5.3: The pose of an IPU U_l is constrained by image correspondences between its primary and secondary cameras C_l^p and C_l^s by forcing the structure of the display surface that is currently observed (blue points) to coincide with the known display surface model.

Remote correspondences provide constraints on an IPU's pose via measurements from other remote IPUs. In this case, correspondences are measured between the primary camera C_l^p of the local IPU U_l and the primary camera C_r^p of another remote IPU U_r. Given an estimate of the pose of C_r^p, this information can be used in conjunction with the known surface geometry to produce constraints on the pose of C_l^p as illustrated in Figure 5.4. Using the estimated pose of C_r^p, each correspondence can be back-projected into a ray that, when intersected with the known display surface model, produces a point on the surface. The resulting set of 3D surface points and their measured 2D positions in C_l^p's image can then be used to produce an estimate of the pose of C_l^p that is conditioned on the pose estimate of C_r^p.

5.2.3 Kalman Filter-Based Estimation

While various geometric algorithms could be used to estimate the pose of an IPU using local and remote correspondences, a Kalman filter [Kalman, 1960, Welch and Bishop, 2006] was chosen for this purpose. There are several advantages to this approach. First, temporal filtering allows the effects of

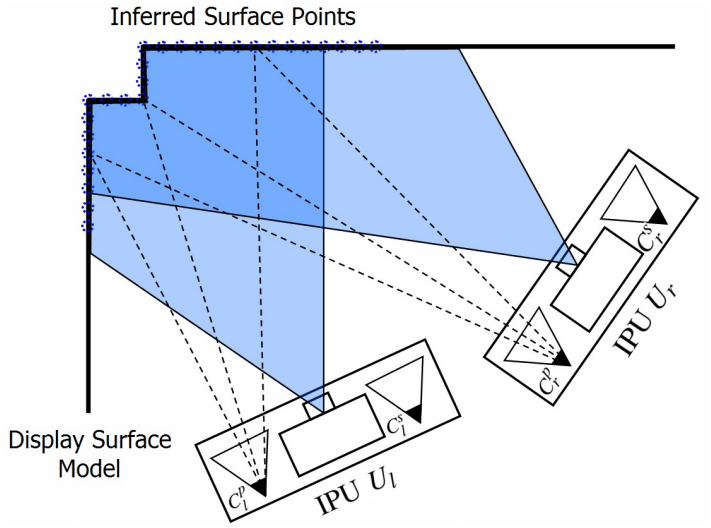


Figure 5.4: The pose estimate of IPU U_r is used as a reference in estimating the pose of IPU U_l via image correspondences between their primary cameras.

measurement noise on pose estimates to be mitigated. Without temporal filtering, measurement noise can cause small variations in the estimated pose over time, ultimately resulting in small changes in the projected imagery that can be distracting to the viewer. Second, with the Kalman filter it is straightforward to account for uncertainty in the pose estimates of remote IPUs for which correspondences have been measured. This is due to the fact that, in addition to estimating the state of the process, the Kalman filter also estimates the state error covariance—an indication of uncertainty in the state estimate due to the failure of measurements to fully constrain a solution.

Filter Operation

Each IPU maintains a Kalman filter that processes the local and remote correspondences obtained at each time step and produces a filtered pose estimate. Due to the non-linear nature of the measurement function, which includes perspective projection, an *extended* Kalman filter is used.

In what follows, superscripts p and s are used to differentiate primary and secondary cameras, while subscripts l and r are used to denote "local" and "remote". Since all IPUs operate identically, consider an arbitrary IPU U_l with primary and secondary cameras C_l^p and C_l^s. Let x_l and P_l be the pose and error covariance estimates of C_l^p and let local correspondences be denoted as $z_l^p \Leftrightarrow z_l^s$, where z_l^p is measured in C_l^p and z_l^s is measured in C_l^s. Additionally, let $U_{r_1}, U_{r_2}, ...U_{r_n}$ be the set of remote IPUs

with primary cameras $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ for which remote correspondences have been measured, and let their respective pose estimates and error covariances be $x_{r_1}, x_{r_2}, ..., x_{r_n}$ and $P_{r_1}, P_{r_2}, ..., P_{r_n}$. Finally, remote correspondences will be denoted as $z^p_{l,r_i} \Leftrightarrow z^p_{r_i,l}$, where z^p_{l,r_i} is a set of feature measurements in local primary camera C^p_l that correspond to a set of feature measurements $z^p_{r_i,l}$ in remote primary camera $C^p_{r_i}$. The notation that will be used in this chapter is summarized in Table 5.1.

The state vector $\widehat{X}_k$ that is estimated by the Kalman filter at each time step k aggregates the pose of C^p_l and the poses of the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$

$$\widehat{X}_k = \begin{bmatrix} x_l \\ x_{r_1} \\ x_{r_2} \\ \vdots \\ x_{r_n} \end{bmatrix}, \tag{5.1}$$

where the "super hat" symbol $\wedge$ is used to denote aggregation.

Formulating the state vector in this way allows the filter to take into account uncertainty in the poses of the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ since their individual error covariances appear in the aggregate error covariance $\widehat{P}_k$ estimated by the filter

$$\widehat{P}_k = \begin{bmatrix} P_l & P_{r_1,l} & P_{r_2,l} & \dots & P_{r_n,l} \\ P_{l,r_1} & P_{r_1} & 0 & \dots & 0 \\ P_{l,r_2} & 0 & P_{r_2} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ P_{l,r_n} & 0 & \dots & 0 & P_{r_n} \end{bmatrix}. \tag{5.2}$$

The individual error covariances of C^p_l and the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ lie on the main diagonal while the cross-covariances relating C^p_l to the $C^p_{r_1}, C^p_{r_2}, ..., C^p_{r_n}$ lie in the first row and column. These cross-covariance terms relating the poses of the local and remote IPUs are important in preventing overconfidence in the IPU pose estimates, especially in cases where the poses may be globally unobservable, e.g. due to the shape of the display surface. Cross-covariances between the remote IPUs are not maintained since they are not directly related to the local IPU's pose or error covariance. These covariance terms

Symbol	Explanation
U_l	The local IPU
U_{r_i}	The ith remote IPU
C_l^p	Primary camera of the local IPU
C_l^s	Secondary camera of the local IPU
$C_{r_i}^p$	Primary camera of the ith remote IPU
x_l	Pose of the local IPU's primary camera
x_{r_i}	Pose of the ith remote IPU's primary camera
P_l	Error covariance of the local IPU
P_{r_i}	Error covariance of the ith remote IPU
Q_l	Process noise covariance of the local IPU
Q_{r_i}	Process noise covariance of the ith remote IPU
h_l	Measurement function for local correspondences
h_{r_i}	Measurement function for remote correspondences to the ith remote IPU
H_l	Jacobian of h_l with respect to x_l
H_{l,r_i}	Jacobian of h_{r_i} with respect to x_l
$H_{r_i,l}$	Jacobian of h_{r_i} with respect to x_{r_i}
z_l^p	Set of measurements in the local IPU's primary camera that correspond to z_l^s
z_l^s	Set of measurements in the local IPU's secondary camera that correspond to z_l^p
z_{l,r_i}^p	Set of measurements in the local IPU's primary camera that correspond to $z_{r_i,l}^p$
$z_{r_i,l}^p$	Set of measurements in the ith remote IPU's primary camera that correspond to z_{l,r_i}^p
$\widetilde{z}_l^p$	Prediction of z_l^p
$\widetilde{z}_l^s$	Prediction of z_l^s
$\widetilde{z}_{l,r_i}^p$	Prediction of z_{l,r_i}^p
$\widehat{Z}_k$	Aggregate measurement vector
$\widetilde{Z}_k$	Aggregate measurement prediction vector
$\widehat{X}_k$	Aggregate Kalman filter state vector
$\widehat{P}_k$	Aggregate error covariance matrix
$\widehat{Q}_k$	Aggregate process noise covariance
$\widehat{R}_k$	Aggregate measurement noise covariance
$\widehat{H}_k$	Aggregate measurement Jacobian
K_k	Kalman gain
κ_l^p	Intrinsics of the local IPU's primary camera
κ_l^s	Intrinsics of the local IPU's secondary camera
$\kappa_{r_i}^p$	Intrinsics of the ith remote IPU's primary camera
Δ_l	Coordinate transformation between the local IPU's primary and secondary cameras
S	Display surface model

Table 5.1: Mathematical Notation

are, however, maintained by the remote IPUs and are indirectly incorporated by the local IPU when remote pose and error covariance information is communicated to the local IPU as described later in Section 5.3.

A Kalman filter acts as a predictor-corrector. At each time step the filter employs a *time update* and *measurement update*. These are described in the following sections.

Time Update

The time update phase is responsible for propagating the filter state $\widehat{X}_{k-1}$ and error covariance $\widehat{P}_{k-1}$ forward in time from the previous time step $k-1$, to produce *a priori* state and error covariance estimates $\widehat{X}_k^-$ and $\widehat{P}_k^-$ at time k. Due to the assumption that the IPUs are normally stationary (Assumption 3 in Section 5.2.1) a "constant" motion model is employed, with the following corresponding time update equations:

$$\widehat{X}_k^- = \widehat{X}_{k-1} \tag{5.3}$$

$$\widehat{P}_k^- = \widehat{P}_{k-1} + \widehat{Q}_k. \tag{5.4}$$

Alternatively, other motion models, such as a position-velocity model or multi-modal Kalman filter, could be employed [Bar-Shalom and Li, 1998].

The *process noise* $\widehat{Q}_k$ added to $\widehat{P}_{k-1}$ models uncertainty due to presumed random variations *between* filter updates. A different process noise matrix for each of the local and remote poses $x_l, x_{r_1}, x_{r_2}, ..., x_{r_n}$ is allowed, but it is assumed that there is no correlation between them. Each IPU estimates its own process noise covariance Q_l using the technique described in [Myers and Tapley, 1976].

$$\widehat{Q}_k = \begin{bmatrix} Q_l & 0 & 0 & \dots & 0 \\ 0 & Q_{r_1} & 0 & \dots & 0 \\ 0 & 0 & Q_{r_2} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \dots & 0 & Q_{r_n} \end{bmatrix}. \tag{5.5}$$

Measurement Update

In the measurement update phase, the measurements $\widehat{Z}_k$ at time k are used in conjunction with a set of computed measurement predictions $\widetilde{Z}_k$ to correct the a priori state $\widehat{X}_k^-$ and error covariance $\widehat{P}_k^-$ estimates into *a posteriori* state $\widehat{X}_k$ and error covariance estimates $\widehat{P}_k$. The measurement update equations also require a measurement noise covariance matrix $\widehat{R}$ and a Jacobian matrix $\widehat{H}_k$ that indicates the sensitivity of the measurements to changes in the state parameters.

The measurement update equations that are used are

$$K_k = \widehat{P}_k^- \widehat{H}_k^T \left(\widehat{H}_k \widehat{P}_k^- \widehat{H}_k^T + \widehat{R} \right)^{-1} \tag{5.6}$$

$$\widehat{X}_k = \widehat{X}_k^- + K_k \left(\widehat{Z}_k - \widetilde{Z}_k \right) \tag{5.7}$$

$$\widehat{P}_k = \left(I - K_k \widehat{H}_k \right) \widehat{P}_k^- . \tag{5.8}$$

Using the Jacobian $\widehat{H}_k$, the measurement covariance $\widehat{R}$, and the a priori error covariance $\widehat{P}_k^-$, the so-called Kalman gain K_k is computed. The Kalman gain is used to weight the measurement residual between $\widehat{Z}_k$ and $\widetilde{Z}_k$ to produce a correction to the a priori state estimate.

The measurement vector $\widehat{Z}_k$ aggregates all measurements in the local primary camera C_l^p into a single measurement vector

$$\widehat{Z}_k = \begin{bmatrix} z_l^p \\ z_{l,r_1}^p \\ z_{l,r_2}^p \\ \vdots \\ z_{l,r_n}^p \end{bmatrix} . \tag{5.9}$$

The measurement prediction $\widetilde{Z}_k$ is generated by a mathematical function that predicts the measured values based on the current a priori state estimate. In the case of local correspondences where z_l^p corresponds to z_l^s, the function h_l is used to produce a prediction $\widetilde{z}_l^p$ of z_l^p using the following parameters

$$\widetilde{z}_l^p = h_l\left(x_l; z_l^s, \kappa_l^p, \kappa_l^s, \Delta_l, S\right), \tag{5.10}$$

where x_l is the pose of C_l^p, κ_l^p and κ_l^s are the intrinsics of C_l^p and C_l^s, Δ_l is the coordinate transformation between C_l^p and C_l^s, and S is the model of the display surface.

The function h_l performs the following operation. The pose x_l of C_l^p is used in addition to κ_l^p, κ_l^s, and Δ_l to produce projection matrices for local cameras C_l^p and C_l^s. Each measurement z_l^s in the local secondary camera is back-projected into a ray using the projection matrix of C_l^s, and these rays are intersected with the surface S to produce a set of 3D surface points. The projection matrix of C_l^p is then applied to each of these 3D surface points to produce a prediction $\widetilde{z}_l^p$ of the measurement in the local primary camera.

In the case of remote correspondences, where z_{l,r_i}^p corresponds to $z_{r_i,l}^p$ for remote IPU U_{r_i}, the measurement function h_{r_i} is used to produce $\widetilde{z}_{l,r_i}^p$ (a prediction of z_{l,r_i}^p)

$$\widetilde{z}_{l,r_i}^p = h_{r_i}\left(x_l, x_{r_i}; z_{r_i,l}^p, \kappa_l^p, \kappa_{r_i}^p, S\right), \tag{5.11}$$

where x_l is the pose of C_l^p and x_{r_i} is the pose of $C_{r_i}^p$, κ_l^p and $\kappa_{r_i}^p$ are the intrinsics of C_l^p and $C_{r_i}^p$, and S is the model of the display surface.

The operation of h_{r_i} is analogous to that of h_l. The poses x_l and x_{r_i} of C_l^p and $C_{r_i}^p$ are used in conjunction with κ_l^p and $\kappa_{r_i}^p$ to produce projection matrices for C_l^p and $C_{r_i}^p$. Each of the $z_{r_i,l}^p$ is back-projected into a ray using the projection matrix of $C_{r_i}^p$, and each of these rays is intersected with the surface S to produce a set of 3D surface points. The projection matrix of C_l^p is then applied to each of these surface points to produce $\widetilde{z}_{l,r_i}^p$.

The vector $\widetilde{Z}_k$ is then

$$\widetilde{Z}_k = \begin{bmatrix} \widetilde{z}_l^p \\ \widetilde{z}_{l,r_1}^p \\ \widetilde{z}_{l,r_2}^p \\ \vdots \\ \widetilde{z}_{l,r_n}^p \end{bmatrix} = \begin{bmatrix} h_l\left(x_l^-; z_l^s, \ldots\right) \\ h_{r_1}\left(x_l^-, x_{r_1}^-; z_{r_1,l}^p, \ldots\right) \\ h_{r_2}\left(x_l^-, x_{r_2}^-; z_{r_2,l}^p, \ldots\right) \\ \vdots \\ h_{r_n}\left(x_l^-, x_{r_n}^-; z_{r_n,l}^p, \ldots\right) \end{bmatrix}. \tag{5.12}$$

The Jacobian matrix $\widehat{H}_k$ indicates the sensitivity of the measurements to changes in the state parameters. Since $\widehat{Z}_k$ is split between local and remote correspondences, so too is $\widehat{H}_k$,

$$\widehat{H}_k = \begin{bmatrix} \frac{\partial h_l(x_l^-; z_l^s, \ldots)}{\partial \widehat{X}} \\ \frac{\partial h_{r_1}(x_l^-, x_{r_1}^-; z_{r_1,l}^p, \ldots)}{\partial \widehat{X}} \\ \frac{\partial h_{r_2}(x_l^-, x_{r_2}^-; z_{r_2,l}^p, \ldots)}{\partial \widehat{X}} \\ \vdots \\ \frac{\partial h_{r_n}(x_l^-, x_{r_n}^-; z_{r_n,l}^p, \ldots)}{\partial \widehat{X}} \end{bmatrix}, \tag{5.13}$$

As a result of the dependence of h_l on only C_l^p's pose and the dependence of h_{r_i} on only the poses of C_l^p and $C_{r_i}^p$, $\widehat{H}_k$ has the following block structure

$$\widehat{H}_k = \begin{bmatrix} H_l & 0 & 0 & \ldots & 0 \\ H_{l,r_1} & H_{r_1,l} & 0 & \ldots & 0 \\ H_{l,r_2} & 0 & H_{r_2,l} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ H_{l,r_n} & 0 & \ldots & 0 & H_{r_n,l} \end{bmatrix}, \tag{5.14}$$

where H_l is the Jacobian of the local measurement function h_l with respect to x_l, and H_{l,r_i} and $H_{r_i,l}$ are the Jacobians of the remote measurement function h_{r_i} with respect to x_l and x_{r_i}, respectively. More formally,

$$H_l = \frac{\partial h_l(x_l^-; z_l^s \ldots)}{\partial x_l} \tag{5.15}$$

$$H_{l,r_i} = \frac{\partial h_{r_i}(x_l^-, x_{r_i}^-; z_{r_i,l}^p, \ldots)}{\partial x_l} \tag{5.16}$$

$$H_{r_i,l} = \frac{\partial h_{r_i}(x_l^-, x_{r_i}^-; z_{r_i,l}^p, \ldots)}{\partial x_{r_i}}. \tag{5.17}$$

The final piece of the measurement update equations to discuss is the measurement noise covariance matrix $\widehat{R}$. It is assumed that measurement noise is constant over time and independent across measurements, but that the noise level in local and remote measurements may differ. The matrix $\widehat{R}$

is then a diagonal matrix where the diagonal entries have value v_l for local measurements and v_r for remote measurements

$$\widehat{R} = \begin{bmatrix} v_l & & & & & & \\ & \ddots & & & & & \\ & & v_l & & & & \\ & & & v_r & & & \\ & & & & \ddots & & \\ & & & & & v_r & \end{bmatrix}. \tag{5.18}$$

Sequential Measurement Processing

A sequential measurement processing or "SCAAT" approach to filtering, as discussed in Chapter 3, can be used to reduce the computational cost associated with performing updates to the Kalman filter. This is largely due to the fact that, when processing a single measurement at a time, the matrix inversion that must occur during the measurement update step involves only a 2×2 matrix. As with any kind of sequential measurement processing approach, it is important to randomize the measurements to eliminate any structure that may be present in their ordering. Also, sequential measurement processing can be much less stable in the face of erroneous measurements, such as false image correspondences, and should be used in conjunction with techniques to improve robustness as described in the following section.

Sequential measurement processing would proceed as illustrated in Algorithm 3. Since all measurements originated at the same time, the time update step of the filter remains unchanged and can occur before any measurements from the current time step are processed. Then, for each measurement, the state $\widehat{X}_k$ and error covariance matrices $\widehat{P}_k$, which remain aggregate, are updated using the measurement Jacobian H_s, the non-aggregate Jacobian for a single measurement. The form of H_s will differ depending on whether the measurement is a local or remote correspondence. For a local correspondence, we have simply $H_s = \begin{bmatrix} H_l & 0_{2,6n} \end{bmatrix}$, while for a remote correspondence to remote IPU U_{r_i}, $H_s = \begin{bmatrix} H_{l,r_i} & 0_{2,6(i-1)} & H_{r_i,l} & 0_{2,6(n-i)} \end{bmatrix}$, where $0_{m,n}$ denotes an $m \times n$ matrix of zeros and n denotes the number of remote IPUs with which measurements are currently being exchanged. Note

that H_s must be recomputed with every measurement that is processed since it is a function of the current state estimate. R_s is a 2×2 diagonal matrix where the value of each diagonal element depends on whether it corresponds to a local or remote correspondence. Finally, $\widetilde{z}_s = h_l\left(x_l; z_c, \kappa_l^p, \kappa_l^s, \Delta_l, S\right)$ or $\widetilde{z}_s = h_{r_i}\left(x_l, x_{r_i}; z_c, \kappa_l^p, \kappa_{r_i}^p, S\right)$, depending on whether z_s is from a local or remote correspondence, and z_c is its correspondence.

Input : State $\widehat{X}_{k-1}$, CovarianceMatrices $\widehat{P}_{k-1}$, $\widehat{Q}_k$, R_s, Measurements $\widehat{Z}_k$
Output: State $\widehat{X}_k$, ErrorCovariance $\widehat{P}_k$

$\widehat{X}_k^- = \widehat{X}_{k-1}$
$\widehat{P}_k^- = \widehat{P}_{k-1} + \widehat{Q}_k$
for each measurement $z_s \in \widehat{Z}_k$ **do**
  H_s = Compute-Jacobian($\widehat{X}_k^-$)
  $\widetilde{z}_s$ = Predict-Measurement($\widehat{X}_k^-$)
  $K_s = \widehat{P}_k^- H_s^T \left(H_s \widehat{P}_k^- H_s^T + R_s\right)^{-1}$
  $\widehat{X}_k^- = \widehat{X}_k^- + K_s\left(z_s - \widetilde{z}_s\right)$
  $\widehat{P}_k^- = \left(I - K_s H_s\right)\widehat{P}_k^-$
end for
$\widehat{X}_k = \widehat{X}_k^-$
$\widehat{P}_k = \widehat{P}_k^-$

Algorithm 3: SEQUENTIAL MEASUREMENT PROCESSING

5.2.4 Robustness

The Kalman filter described in this chapter operates under the assumption that the correspondences provided to it are correct, or that the corresponding points in two camera views originate from the same 3D scene point. Unfortunately, *false correspondences* are likely to be encountered in practice since no feature matching algorithm has been shown to produce 100% correct correspondences in all cases. Processing these false correspondences can lead to incorrect estimation and/or instability especially in the case of a non-linear measurement model, as in this distributed cooperative framework.

It is thus necessary to employ some method of identifying and eliminating false correspondences. This becomes even more important when processing measurements sequentially since a single erroneous correspondence can have a drastic effect on the estimation. Fortunately, this is a problem the computer vision community has dealt with for some time and a number of approaches for identifying false correspondences have been developed. For example, [Clipp et al., 2007] proposes a pipelined

Kalman filter approach where multiple filters are offset in time such that the final filter in the pipeline processes only the most reliable measurements. Alternatively, a RANSAC approach based on the constraints of two-view geometry is also possible.

Recall from Chapter 3 two algebraic representations of two-view geometry, the homography and the fundamental matrix, both of which can be estimated from image correspondences between two views and then used to validate further correspondences. Computation of both these entities can also be made robust through the use of RANSAC, which will directly determine inlying and outlying image correspondences. Thus, given a set of putative image correspondences between two views, a RANSAC-based computation of a homography or fundamental matrix can be used to divide the correspondences into sets of inliers and outliers. The false correspondences are discarded as outliers and the inliers are forwarded to the Kalman filter for processing.

Which of these algebraic entities to use, a homography or the fundamental matrix, depends largely on the type of display surface that is being used, or is expected to be used. In displays consisting of a planar or multi-planar display surface, a homography-based approach is most appropriate since computation of the fundamental matrix is degenerate in this case. In the case of a curved or more general surface geometry, a RANSAC-based computation of the fundamental matrix should be used.

5.3 Implementation

This section describes the distributed system that realizes the computational framework described in Section 5.2.

5.3.1 Pre-Calibration

Before system operation, the internal calibration of each IPU and the model of the display surface must be estimated in addition to obtaining an initial estimate of each IPU's pose.

The process used to estimate the internal calibration of each IPU consists of first calibrating the IPU's stereo camera pair using the Matlab Camera Calibration Toolbox [Bouguet, 2008]. Once the cameras have been calibrated, the projector calibration is estimated by projecting structured light patterns onto a non-planar surface and capturing the projected patterns with the IPU's cameras. The

resulting images are then decoded to produce three-way image correspondences between the cameras and the projector. The correspondences between the cameras are then triangulated into 3D points to produce a set of 3D-2D correspondences in the projector that is then used to calibrate the projector using the DLT algorithm [Abdel-Aziz and Karara, 1971].

Once the internal calibration of each IPU has been estimated, they are arranged to form a display, and the display surface geometry and initial pose of each IPU is estimated. This pre-calibration process assumes that the camera field-of-view of each IPU overlaps with the camera field-of-view of at least one other IPU. The IPUs then take turns projecting *encoded* structured light patterns while the cameras of all IPUs capture images. Decoding of these structured light patterns allows precise inter- and intra-IPU image correspondences to be obtained.

The intra-IPU correspondences are used to reconstruct a point-cloud representation of the display surface from the perspective of each IPU. The inter-IPU correspondences are then used to stitch these individual reconstructions together, resulting in a point-cloud representation of the display surface and an estimate of the pose of each IPU together in a common coordinate system.

To construct a polygonal model from this point-cloud representation, a RANSAC-based plane-fitting algorithm described in [Quirk et al., 2006] is used, which is robust against noise and outlying points resulting from false stereo matching. This algorithm extracts planes from the point cloud representation of the display surface and intersects them to produce a polygonal model of the surface.

5.3.2 Distributed Architecture

In order to continuously estimate its pose, each IPU must collect local and remote correspondences and process them using its Kalman filter. While local correspondences can be collected at each IPU without the need to communicate with other IPUs, a mechanism for obtaining remote correspondences is required. The system that has been developed accomplishes this through inter-IPU communication of camera images over a local network. An alternative approach that favors communicating feature vectors instead of camera images is described in Section 5.5.5.

A request/response architecture is used to allow IPUs to operate asynchronously by requesting primary camera images from other IPUs captured at a specified time. Camera synchronization hardware from Point Grey Research (http://www.ptgrey.com) is currently used to synchronize the exposures of

all cameras and provide an absolute time reference between IPUs.

To facilitate the ability of IPUs to respond to requests for camera images captured at a specified time, each IPU maintains a history of recently captured camera images in its *camera buffer*. Each IPU maintains two camera buffers, one each for its primary and secondary cameras. A special camera buffer thread is dedicated to updating the camera buffer by replacing the oldest image in the buffer with a new image from the camera whenever one is available. In this way, a recent history of camera images is available at each IPU that can be searched based on time stamp when an image request is received from another IPU.

Collection and Processing of Local Correspondences

The following process occurs asynchronously at each IPU to collect a set of local correspondences. First, the latest image is requested from the camera buffer of the local primary camera, call this image I_l^p. Once this image has been obtained, the corresponding image in time I_l^s from the camera buffer of the local secondary camera is requested.

The next step is to obtain correspondences between I_l^p and I_l^s. This is done by first detecting a set of features in I_l^p using the OpenCV implementation of [Shi and Tomasi, 1994]. Correspondences for these features are then found in I_l^s using the OpenCV implementation of KLT tracking [Lucas and Kanade, 1981, Bouguet, 1999]. Finally, each correspondence is checked against the epipolar constraint between the primary and secondary cameras to yield the $z_l^p \Leftrightarrow z_l^s$ from Section 5.2.

Each IPU is provided with local access to its own fixed internal calibration as well as the display surface model, which is currently assumed to be static but will be allowed to change in Chapter 6. In conjunction with the $z_l^p \Leftrightarrow z_l^s$, this provides all necessary information to compute the Jacobian H_l from Equation 5.15 as well as $\widetilde{z}_l^p$ using Equation 5.10. The Jacobian H_l can be estimated numerically using forward differences.

Collection and Processing of Remote Correspondences

Remote correspondences are collected by requesting camera images from other IPUs that were captured by their primary cameras at the same time as I_l^p. When such a request is processed by another

IPU U_{r_i}, its response includes not only the requested camera image $I^p_{r_i}$, but also its intrinsics $\kappa^p_{r_i}$, current pose estimate x_{r_i}, and a priori error covariance $(P_{r_i} + Q_{r_i})$. In using the current pose estimate rather than the pose estimate at the time $I^p_{r_i}$ was captured, it is effectively assumed that the time that has elapsed between when $I^p_{r_i}$ was captured and when it was requested in small, or that any changes in pose during this time are small. Should this assumption be violated in practice, it possible to maintain a history of pose and error covariance estimates along with the history of camera images in the camera buffer.

Once the requested image $I^p_{r_i}$ has been received from another IPU, correspondences between I^p_l and $I^p_{r_i}$ are measured. This is accomplished by first finding a set of correspondences between $I^p_{r_i}$ and $\widetilde{I}^p_{r_i}$, a prediction of $I^p_{r_i}$ generated on graphics hardware using the current estimated calibration of C^p_l and $C^p_{r_i}$, and then transforming these into a set of correspondences between I^p_l and $I^p_{r_i}$. This approach greatly improves feature matching success for algorithms like KLT when there are large perspective distortions between the two views, as is likely the case for camera images from different IPUs. More details on this technique can be found in [Johnson and Fuchs, 2007a], which is included in the Appendix.

Once the correspondences $z^p_{l,r_i} \Leftrightarrow z^p_{r_i,l}$, between I^p_l and $I^p_{r_i}$ have been measured, all information necessary to compute H_{l,r_i}, $H_{r_i,l}$, and $\widetilde{z}^p_{l,r_i}$ from Equations 5.16, 5.17, and 5.11 is available. The Jacobian H_{l,r_i} is computed using the closed-form solution in [Haralick and Shapiro, 1993], and the Jacobian $H_{r_i,l}$ is estimated numerically using forward differences.

Continuous Operation

Algorithms 5 and 6 summarize the implementation for collecting and processing local and remote correspondences and Algorithm 7 describes the steps involved in processing a request received from a remote IPU. Algorithm 4 illustrates how these processes are organized into a continuous calibration loop that is executed by each IPU at display time. This loop is executed independently of the application rendering in a separate calibration thread. This calibration thread is implemented to run concurrently with the rendering thread, but without causing rendering performance to drop below a certain frame rate.

In the current implementation, each IPU broadcasts its image requests to all IPUs in the display

and processes each of their responses as they are received. While this limits the scalability of the system, plans to improve this are described in Section 5.5.3.

In order to absorb network latency, each IPU collects and processes local correspondences while it waits to receive image responses from the other IPUs. It then enters an inner loop where it processes camera image responses and requests until all IPUs have responded or a timeout condition has been reached. This timeout condition provides fault tolerance by allowing system operation to continue when an IPU is unable to provide a response for some reason.

The final step is to update the Kalman filter to produce a new pose estimate that is communicated to the rendering process in order to allow the new estimate to affect the image correction that takes place.

Input : Initial display calibration incl. DisplaySurface S, Intrinsics κ_l^p,κ_l^s, Extrinsics x_l, and CoordinateTransform Δ_l
Output: Continuous estimate of local IPU pose x_l

1: **while** true **do**
2: $[I_l^p, I_l^s]$ = Get-Local-Camera-Images
3: BroadCast-Request(I_l^p.time)
4: Process-Local
5: **repeat**
6: **if** *responseReceived* **then**
7: Process-Remote;
8: **end if**
9: **if** *requestReceived* **then**
10: Process-Request
11: **end if**
12: **until** $timeout \vee allResponsesReceived$
13: Update-Kalman-Filter
14: **end while**

Algorithm 4: CONTINUOUSPOSEESTIMATION

Input : Image I_l^p,I_l^s, Intrinsics κ_l^p,κ_l^s, Extrinsics x_l, CoordinateTransform Δ_l, DisplaySurface S
Output: ImageMeasurements z_l^p, z_l^s, MeasurementJacobian H_l, PredictedMeasurements $\tilde{z}_l^p$

1: z_l^p = Detect-Features(I_l^p)
2: z_l^s = Match-Features(z_l^p,I_l^p,I_l^s)
3: $[z_l^p, z_l^s]$ = Verify-Epipolar-Constraint(z_l^p,z_l^s, κ_l^p,κ_l^s,x_l,Δ_l)
4: $[H_l, \tilde{z}_l^p]$ = Compute-Filter-Mats-L(x_l,z_l^s,κ_l^p,κ_l^s,Δ_l,S)
5: Add-Local-Correspondences-To-Filter(z_l^p,z_l^s,H_l,$\tilde{z}_l^p$)

Algorithm 5: PROCESS-LOCAL

Input : Image $I_l^p, I_{r_i}^p$, Intrinsics $\kappa_l^p, \kappa_{r_i}^p$, Extrinsics x_l, x_{r_i}, DisplaySurface S
Output: ImageMeasurements $z_{l,r_i}^p, z_{r_i,l}^p$, MeasurementJacobians H_{l,r_i}, $H_{r_i,l}$, PredictedMeasurements $\widetilde{z}_{l,r_i}^p$

1: $\widetilde{I}_{r_i}^p$ = Predict-Remote-Image(I_l^p, $\kappa_l^p, \kappa_{r_i}^p, x_l, x_{r_i}, S$)
2: $\widetilde{F}$ = Detect-Features($\widetilde{I}_{r_i}^p$)
3: $z_{r_i,l}^p$ = Match-Features($\widetilde{F}, \widetilde{I}_{r_i}^p, I_{r_i}^p$)
4: z_{l,r_i}^p = Warp-Features($\widetilde{F}$, $\kappa_l^p, \kappa_{r_i}^p, x_l, x_{r_i}, S$)
5: $[H_{l,r_i}, H_{r_i,l}, \widetilde{z}_{l,r_i}^p]$ = Compute-Filter-Mats-R($x_l, x_{r_i}, z_{r_i,l}^p, \kappa_l^p, \kappa_{r_i}^p, S$)
6: Add-Remote-Correspondences-To-Filter($z_{l,r_i}^p, z_{r_i,l}^p, H_{l,r_i}$, $H_{r_i,l}$, $\widetilde{z}_{l,r_i}^p$)

Algorithm 6: PROCESS-REMOTE

Input : Intrinsics κ_l^p, Extrinsics x_l, Covariances P_l, Q_l, Time-Stamp t
Output: Data response sent to remote host

1: I = Search-Camera-Buffer(t)
2: Send-Response($I, \kappa_l^p, x_l, P_l + Q_l$)

Algorithm 7: PROCESS-REQUEST

5.4 Results

The framework for distributed cooperative pose estimation described in this chapter was tested using two real-time applications in two and three-IPU configurations. The first application displays a rotating panorama of real-world imagery that contains many strong features, while the second application is an open source flight simulator called Flight Gear, whose synthetic imagery contains far fewer features. Figure 5.5 shows the capability of the system to continuously estimate the pose of two IPUs when both are moved simultaneously using the panorama application. Similar results for the flight simulator application are shown in Figure 5.6. Figure 5.7 shows a close-up of the projector image overlap as the projectors are moved and recalibrated. Since the framework does not currently support dynamic recomputation of intensity blending masks, no photometric compensation is performed, resulting in the bright bands visible in the images where the projectors overlap.

Figure 5.8 shows the results of pose estimation over time for one IPU in a two-IPU display using the panorama application. In this sequence, captured over roughly four minutes, the IPU is moved once about halfway through the sequence. Frames extracted from the corresponding video sequence show the display configuration before, during, and after movement of the projector. Rotation of the panorama was disabled during this experiment for comparison purposes. The ringing effect in the

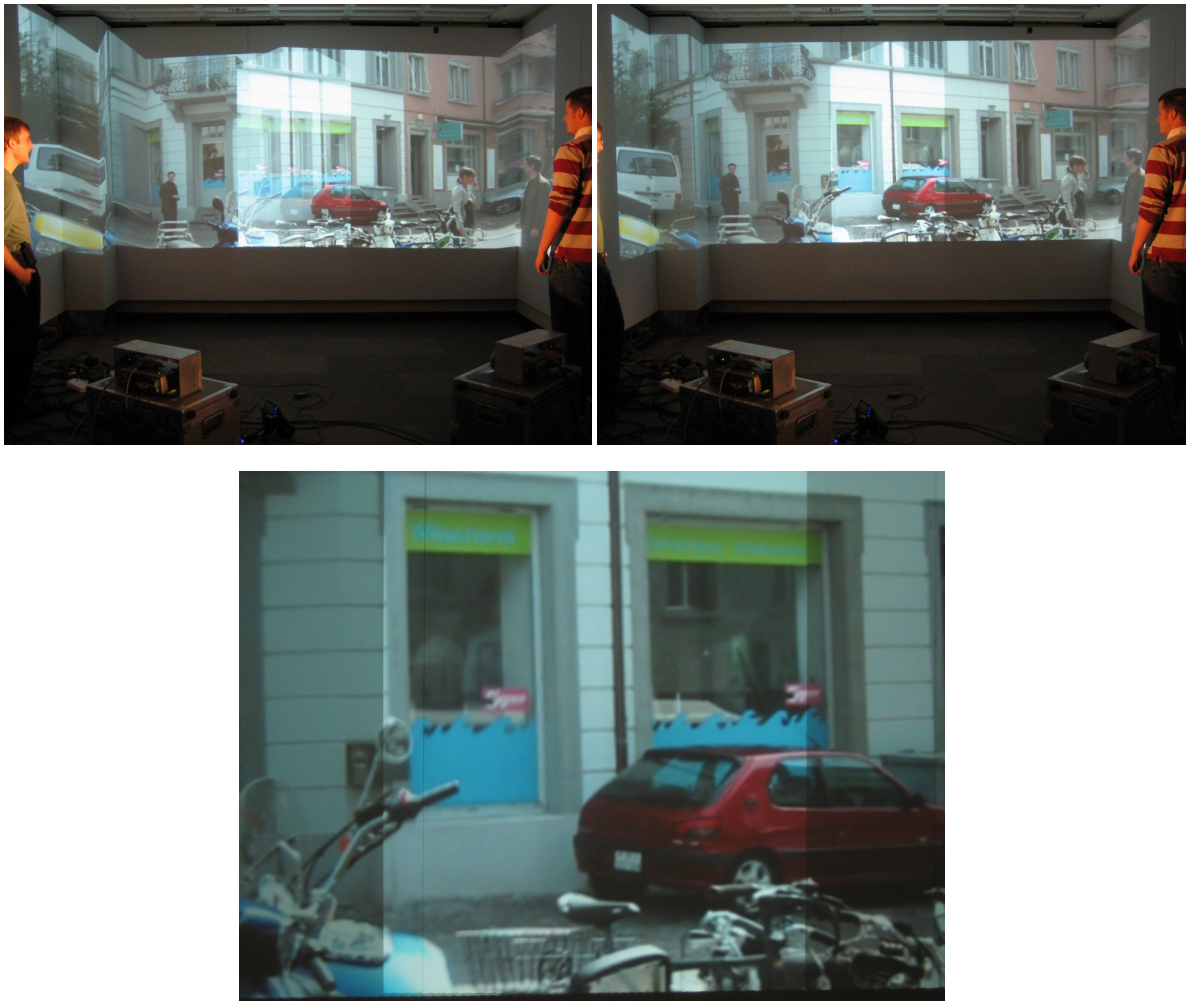


Figure 5.5: A two-projector display after both projectors have been moved (left) and after roughly ten seconds of continuous calibration (right). A close-up of the corrected imagery is shown in the lower image to illustrate the accuracy of the image registration. No photometric correction is being performed, resulting in the clearly visible projector overlap region in roughly the center of the display.

plots as the IPU is moved is a result of the temporary violation of the assumption inherent in the motion model that the IPU is stationary. Also, a slight drift over time may be observed in the y component of the IPU's position. Due to the vertical ambiguity in the shape of the display surface, which corresponds to the y axis, the y component of projector pose was unconstrained in the experimental set-up.

To obtain a quantifiable comparison of the poses estimated using this distributed cooperative framework to actual ground truth, one IPU in a two-IPU display was placed on a checkerboard pattern with checkers of known size and moved through a series of known displacements. The experimental

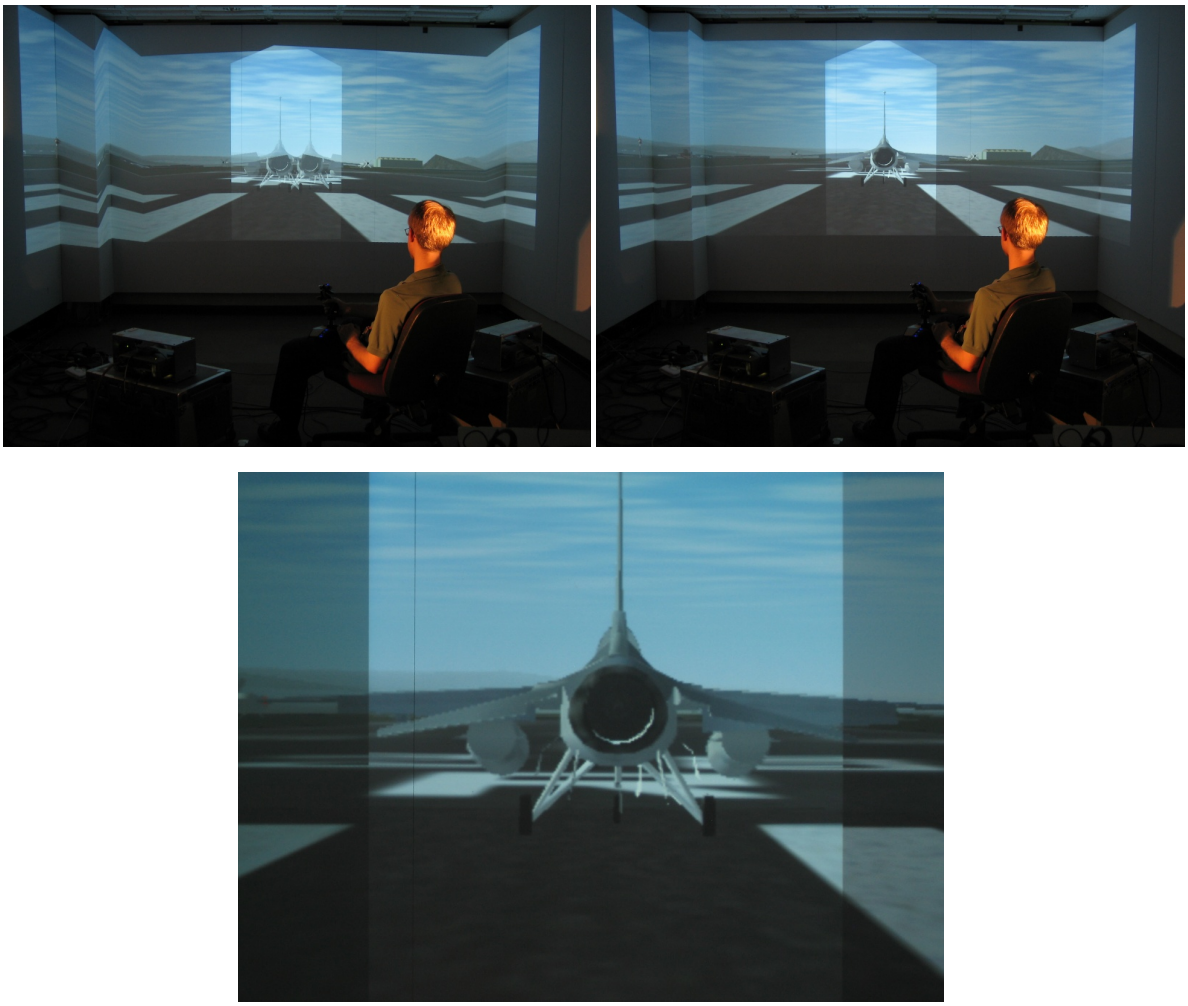


Figure 5.6: A two-projector display after both projectors have been moved (left) and after roughly ten seconds of continuous calibration (right). A close-up of the corrected imagery is shown in the lower image to illustrate the accuracy of the image registration. No photometric correction is being performed, resulting in the clearly visible projector overlap region in roughly the center of the display.

Figure 5.7: A close-up of calibration accuracy in projector image overlap after both projectors have been moved (left) and after roughly 10 seconds of projector pose refinement (right).

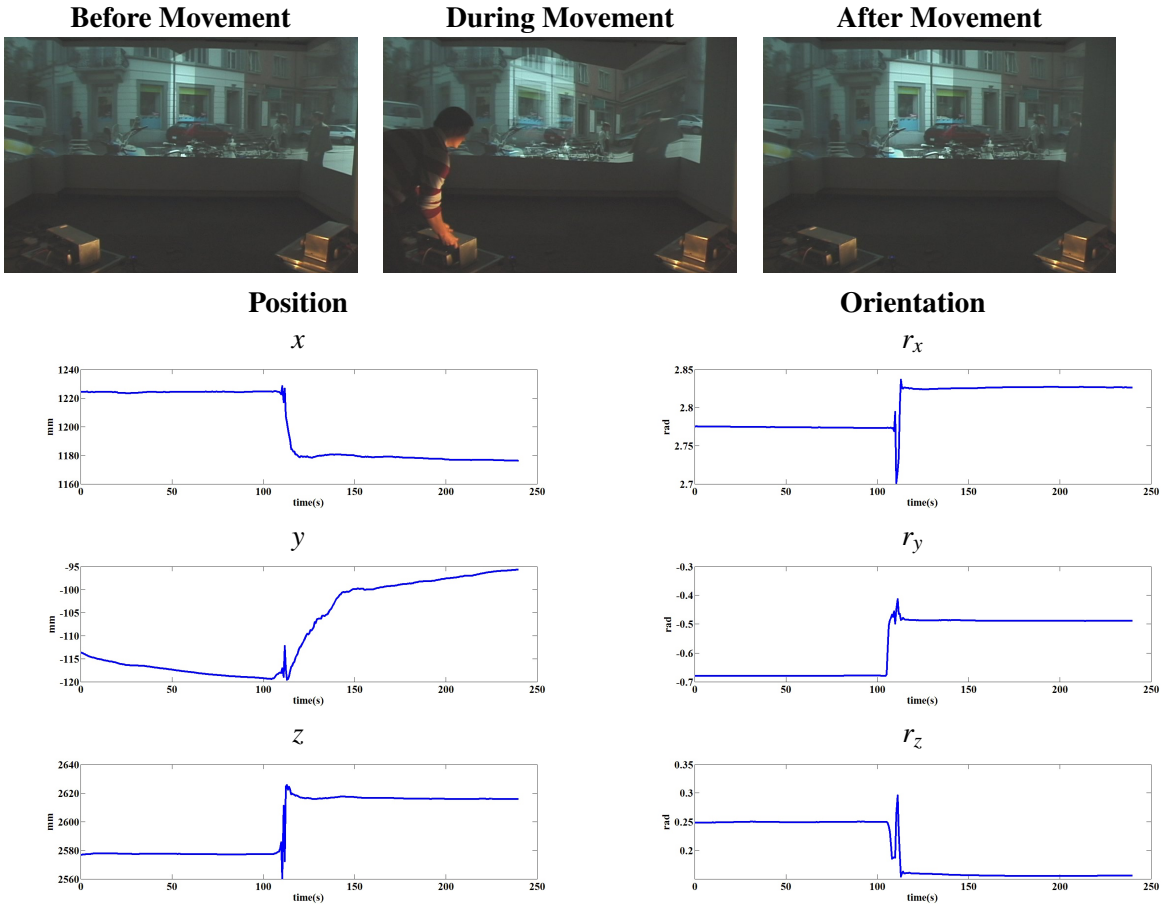


Figure 5.8: Results of estimating the pose of a single projector as it is moved once over the course of four minutes. Frames from the corresponding video sequence show the configuration of the display before, during, and after the projector is moved.

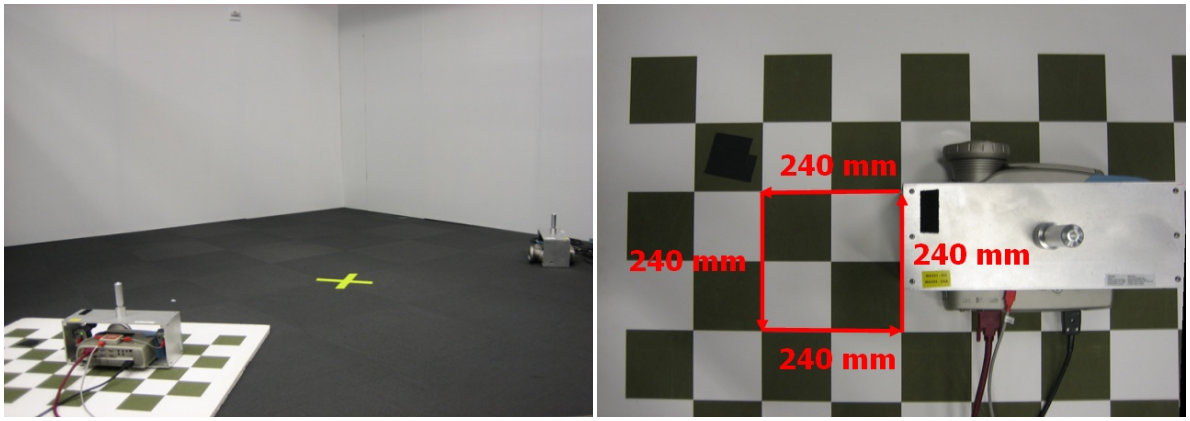


Figure 5.9: Experimental set-up for comparison to ground truth. One IPU is placed on a checkerboard pattern and moved through a series of known displacements as its pose is estimated.

set-up is shown in Figure 5.9, where the IPU was moved through four displacements in the pattern of a square with sides of length 240mm. After each displacement of approximately 240mm, the IPU remained stationary for approximately one minute, resulting in an experiment that lasted roughly five minutes.

The plane of the checkerboard pattern in this experiment corresponded to the x-z plane in calibration coordinates. Plots of the x and z position estimates across time are given in Figure 5.10. The plots clearly show the changes in the estimated position of the IPU as it is moved. The points P1-P4 in the image have been selected to represent the position estimate of the filter just before the IPU is moved. These points are shown connected by solid red lines in the plot of the estimated z versus x positions in Figure 5.11 left. The estimated displacement between each pair of consecutive points is also shown superimposed on the graph. As can be seen, the estimated displacements are close to the ground truth of 240mm. The estimated position of the IPU appears to "curve" as it is moved between positions. This curved pattern to the estimates is a result of dynamically estimating the process noise in the IPU's pose. In this case, a different amount of uncertainty or process noise is being added to each of the parameters, allowing the filter to make larger changes to the parameter with larger process noise and resulting in faster convergence.

Figure 5.11 right shows a plot of absolute distance from the point P1 across time. After the first displacement, the distance approaches the ground truth distance of 240mm. After the second displacement, the distance increases further to approach the ground truth of 339.4mm (the length of

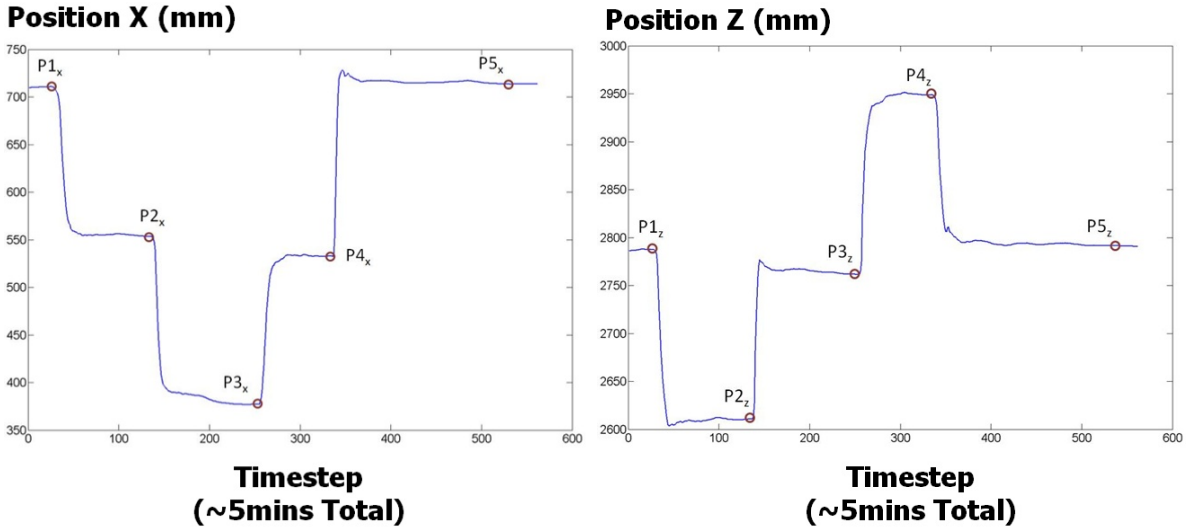


Figure 5.10: Plots of x and z position estimates across time as the IPU is moved through four known displacements. The points P1-P4 have been selected to represent the position estimate of the filter just before the IPU is moved. The point P5 represents the estimate of filter after the IPU is returned to its initial position.

the hypotenuse of the motion). The estimated displacements for the final two displacements are close to the ground truth as well. It should be noted that since the IPU was placed manually by a human in these experiments, human error on the order of a few millimeters is likely.

5.5 Discussion

This section discusses several topics related to the distributed cooperative framework that has just been described including observability, the incorporation of additional measurement types, and scalability.

5.5.1 Observability

It is possible that the configuration of features that are measured cannot fully constrain the pose of one or more projectors. There is such an ambiguity in the example of Figure 5.8 where, due to the shape of the display surface (which included only vertical walls), the vertical component of projector location is unconstrained. This is because the display surface can be thought of as a vertically-extruded surface such that the measurement residuals are unaffected as the IPU is moved along this vertical (y) direction. A general test for observability using the *observability matrix* [Grewal and Andrews, 2008]

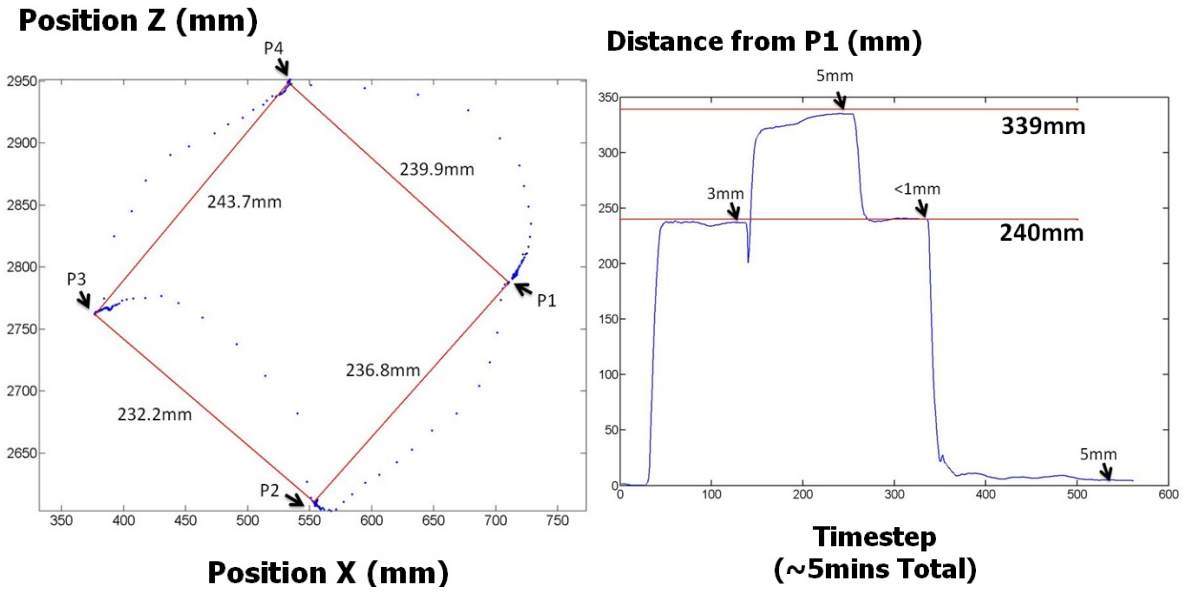


Figure 5.11: A plot of z versus x position estimates with points P1-P4 from Figure 5.10 connected by red lines (left). The estimated displacement between consecutive points from P1-P4 is also shown. A plot of absolute distance from P1 across time (right).

is described in Chapter 3. In the case of the Kalman filter that is used in this framework, which is time-invariant and uses a contant motion model, the observability test simplifies to evaluating the rank of the measurement Jacobian $\widehat{H}_k$, where the system is observable if $\widehat{H}_k$ has row rank equal to the number of state parameters in the state vector $\widehat{X}_k$.

In the distributed cooperative framework that has been presented, when the pose of at least one IPU is constrained in a direction that may be unobservable for others, this will be reflected in its error covariance matrix and will allow the IPU to propagate constraints to other IPUs via remote correspondences. Also, in the situation where the poses of all IPUs are unconstrained in some direction, as in the examples above, the behavior of framework is for the IPUs to estimate values that result in their projected imagery being registered together. While this is a globally consistent and visually appealing result, the result may not be consistent with the viewing position, resulting in an inability of the system to correctly compensate for the shape of the display surface. This could be remedied by continuously estimating the viewer's position with respect to one of the projectors, adding a fiducial to the display surface whose position is measured continuously, or adding a horizontal floor or ceiling plane to the display surface.

5.5.2 Additional Image Measurement Types

The framework, as described in this chapter, uses 2D feature points detected in camera images as measurements. While 2D feature points are likely to be useful in a variety of application, one can imagine situations where other types of measurements may be more desirable. For example the projected imagery may be rich in lines that could be detected and matched between camera images. It may also be desirable to combine measurement types together, e.g. points and lines, when performing the estimation. It is simple to extend the framework to handle such measurement strategies, and the case of points and lines is described here.

Akin to points, lines have two degrees of freedom and each line measurement in an image can be represented by two scalar values (the choice of line representation is not important as long as it is used consistently). From a representation perspective this means there is no difference between points and lines, and we can interchange them in the measurement vector $\widehat{Z}_k$. However, we must be careful to distinguish between points and lines when it comes to the measurement functions h_l and h_{r_i} that are used to process them, which in turn affects computation of $\widetilde{Z}_k$ and $\widehat{H}_k$ from Equations 5.12 and 5.13.

The measurement function for local line correspondences should act as follows. The pose x_l of C_l^p is used in addition to κ_l^p, κ_l^s, and Δ_l to produce projection matrices for C_l^p and C_l^s. Each line correspondence is back-projected into a *plane* using the projection matrix of C_l^s and intersected with the surface S to produce a line on the surface. In general, the intersection of a plane and a surface will be a curve, however the fact that we have corresponding lines between two images guarantees that the intersection results in a line on the surface. This is because the corresponding lines in the two camera images can each be back-projected into a plane, and the intersection of two planes forms a line in non-degenerate cases. Finally, the projection matrix of C_l^p is then applied to each of these lines on the surface to produce predicted line $\widetilde{z}_l^p$ measurements in the primary camera. The measurement function for remote line correspondences follows directly from this. Since $\widetilde{Z}_k$ and $\widehat{H}_k$ are stacked matrices (according to measurements), we can simply select the appropriate measurement function for each measurement on a case-by-case basis when computing them.

In general, any type of image measurement is compatible with this framework as long as its mapping between two camera views (prediction) and the sensitivity of this mapping to changes in the

state parameters can be modeled.

5.5.3 Scalability

There are several issues that may affect the scalability, in terms of the number of IPUs, of an implementation of the distributed cooperative framework described in this chapter. First of all, there is the consideration of how increasing the number of IPUs affects the cost of the Kalman filter update that each IPU performs locally to estimate its own pose and, secondly, the cost of obtaining the 2D image correspondences that are the input to the filter.

The computational complexity of updating the Kalman filter that is used to process local and remote correspondences into an estimate of each IPU's pose is considered first. When processing measurements in batch, the measurement update of the filter, which is performed by each IPU locally, requires $O(m^3 + n^3)$ time (assuming cubic-time matrix multiplication and inversion), where m is the total number of measurements (local and remote) processed at the current time step, and n is one plus the number of remote IPUs for which remote correspondences have been measured. In practice, it is reasonable to assume $m >> n$ since it is desirable to obtain a large number of measurements in each time step in order to amortize the cost of distributed communication. When measurements are processed sequentially, the complexity of the time update step becomes $O(mn^3)$ since the processing of each measurement involves updating the error covariance matrix via matrix multiplication. Even in a large display with many IPUs, it is likely that the cameras of each IPU will share an overlapping field-of-view with only a small number of other IPUs. It is thus reasonable to assume n is small, or that each IPU shares measurements with only a small number of other IPUs, resulting in a significant reduction in computational complexity when processing measurements sequentially.

In performing the filter updates there may also be a hidden cost associated with computing the measurement prediction (Equations 5.10 and 5.11) depending on the type of surface representation. For example, given a polygonal mesh representation, forming the measurement prediction requires that a ray be intersected with the display surface geometry once for each measurement. For dense surfaces with many faces, this can be a costly operation and should be combined with efficient techniques for ray casting, such as the use of a k-d tree.

Secondly, there is the cost of obtaining the image correspondences that are the input to the filter.

This involves the cost of communicating camera images, or possibly just feature vectors, across the network, as well as the cost of feature detection and matching in camera images. Since an IPU with no camera overlap cannot contribute any remote measurements to a local IPU, the scalability of an implementation of the framework can be improved by limiting the number of IPUs that each IPU communicates with to just those whose cameras have an overlapping field-of-view.

5.5.4 Sequential versus Batch Processing

While a sequential measurement processing paradigm has been advocated in this chapter primarily for the sake of computational efficiency, it is of course possible to reach a desired balance between computation time and robustness to outliers by adjusting the number of measurements processed at once from a single measurement up to the total number of measurements that have been obtained at each time step. In the end, however, a single-measurement-at-a-time approach is likely to be best since it allows time to collect additional features that could be chosen as part of a measurement selection strategy designed to reduce uncertainty along directions in the state space where uncertainty is currently highest, as described in Section 3.4.1.

5.5.5 Choice of Feature Matching Approach

As described in the Implementation Section 5.3, the KLT tracking algorithm is used in conjunction with perspective warping of remote camera images to obtain correspondences between cameras, in effect using a feature *tracking* algorithm to solve a feature *matching* problem. This approach was chosen mainly due to its robustness against large differences in perspective between the cameras of IPUs placed far apart, but alternative solutions are not without their advantages. For example, matching SIFT [Lowe, 2004] features would allow the communication of entire camera images across the network to be replaced with the more efficient communication of feature vectors and locations.

5.6 Summary

This chapter has presented a novel distributed calibration framework for multi-projector displays where intelligent projector units interact to cooperatively re-estimate the poses of all projectors during

actual display use. By making use of features in the projected imagery itself, this technique can be applied to any type of projector and is able to operate without altering the projected imagery or affecting its quality. The following chapter describes how this can be extended to estimate information about the display surface as well.

While local and remote correspondences could be obtained without the existence of a secondary camera mounted to each IPU, for example by substituting measurements in the secondary camera for measurements in the projector's image, the use of the secondary camera allows each IPU to obtain local measurements in imagery that may not have originated from its own projector. This has the potential to greatly increase the number of available measurements in practice.

It is possible for calibration accuracy to be quite poor for some applications where the imagery is so lacking in strong features that calibration may not be successful. In this case, since the computational framework imposes no requirements on the source of the image measurements, it could be used in conjunction with techniques for embedding imperceptible patterns into projected imagery [Cotting et al., 2004, Cotting et al., 2005, Grundhöfer et al., 2007].

The framework presented in this chapter can impose a significant amount of computational overhead that competes with the rendering process for resources when both operate on the same machine. These performance penalties can be overcome by fully integrating a computational unit with each IPU whose sole responsibility is to estimate the unit's pose. The rendering application could then operate on a separate machine that periodically receives updates of IPU's current pose.

CHAPTER 6

EXTENSIONS FOR CONTINUOUS DISPLAY SURFACE ESTIMATION

In the previous chapter, a computational framework was presented that allows a group of intelligent projector units (IPUs) to cooperatively estimate the poses of all IPUs in a continuous manner via communication over a local network. The formulation of the framework that was presented does, however, rely on one assumption that may be undesirable in certain situations: the assumption that the geometry of the display surface is known and static. This limits the usefulness of the framework in situations where the parameters of the display surface, such as its position or shape, may change over time. Also, even in cases where the display surface is static, the model of the surface may contain small errors that should be corrected in order to achieve the best possible geometric image correction.

This chapter describes how the distributed cooperative framework for continuous calibration from Chapter 5 can be extended to estimate the poses of all projectors and information about the display surface, such as its shape or pose, in a unified fashion. A description of how the distributed cooperative framework can be extended to estimate parameters of the display surface in two practical scenarios is provided. The first scenario describes a dynamic shader lamps display [Bandyopadhyay et al., 2001, Raskar et al., 2001] where the pose of a rigid, moving display surface is estimated by the framework in addition to the poses of all projectors. The second describes how a polygonal mesh representation of a multi-planar display surface can be refined in a particularly important situation to improve alignment of mesh edges with edges of the physical display surface.

6.1 Unified Projector Pose and Display Surface Estimation

In general, information about the display surface can be estimated using the framework of the previous chapter by including it in the Kalman filter state vector. In order to do so, however, the information about the display surface we wish to estimate must first be *parameterized*, or represented as a set of variables x_s, which will be referred to collectively as the *display surface state vector*.

6.1.1 Display Surface State Parameterization

Since the information about the display surface that must be estimated is likely to vary widely between displays and applications, some examples of appropriate parameterizations in a variety of practical scenarios are provided here.

Dynamic Shader Lamps

Consider a dynamic shader lamps display [Bandyopadhyay et al., 2001] where an architectural model augmented with projected texture (Figure 6.1) can be moved by the viewer. We might assume that the geometry of the display surface is static and known, but that the surface as a whole may move. For example, the surface might be located on a turn table that can be rotated by the user, requiring us to estimate the angle r of the surface with respect to its axis of rotation. In this case, the display surface state vector would simply be $x_s = [r]$. Alternatively, we might not be able to place any constraints on the motion of the surface, requiring us to represent the full six degrees of freedom in its pose. The display surface state vector would then be $x_s = [\ S_x \quad S_y \quad S_z \quad S_\psi \quad S_\theta \quad S_\phi\ ]^T$, with (S_x, S_y, S_z) the location of the display surface and $(S_\psi, S_\theta, S_\phi)$ its orientation, which is the same as the chosen representation of projector pose.

Estimating and Refining Surface Shape

In other examples, we might expect the shape of the display surface to change, requiring us to parameterize the surface shape in some way. For example, if the display surface consists of two walls that intersect in a corner, we might choose to represent each wall as an infinite plane $l_i = [\ x \quad y \quad z \quad d\ ]^T$,

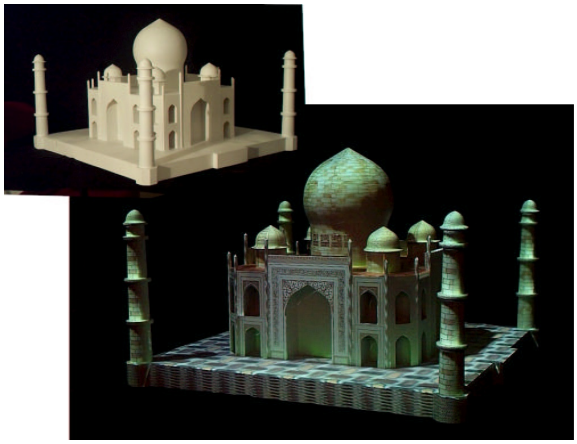


Figure 6.1: A shader lamps display. An untextured architectural model is augmented with projected texture [Raskar et al., 2001].

where (x, y, z) is the plane normal and d is its offset. In this case, the display surface state vector would be $x_s = [\ l_1 \quad l_2\ ]^T$.

In some situations, the surface shape may be very complex and/or difficult to represent using commonly used parametric shapes such as planes, cylinders, and curves. In this case, it may be appropriate to represent the surface as a polygonal mesh or an unorganized set of points. This representation is described further in the Section 6.5.2 in the context of refining the shape of the display surface as it is represented by a polygonal mesh.

Remarks

When choosing how to parameterize the information about the display surface that is to be estimated by the filter, such as its shape and/or orientation, it is desirable to choose a concise representation that accurately reflects any constraints on the shape of the surface. For example, it is superior to represent two planar walls intersecting in a corner as two infinite planes rather than a polygonal mesh of two quads sharing a pair of vertices. The more compact representation of two infinite planes is not only more computationally efficient from a filtering standpoint due to the fewer number of parameters, but will also exhibit greater estimation stability since the mesh representation has many weakly observable parameters. For example, each quad is essentially representing a plane using twelve parameters (three parameters for each of its four vertices). The over-parameterization of the problem can lead to issues with convergence (optimizing in a 12D space as opposed to 4D) and drift, since not all parameters

will be strongly observable.

6.1.2 Kalman Filter Modifications

Regardless of the parameterization of the display surface state, incorporating the estimation of x_s into the Kalman filter involves straightforward modifications to the Kalman filter state vector, error covariance matrix, and measurement Jacobian. Incorporating the display surface state x_s into the Kalman filter state vector, Equation 5.1 from the previous chapter becomes

$$\widehat{X}_k = \begin{bmatrix} \widehat{X}_I \\ x_S \end{bmatrix}, \quad \widehat{X}_I = \begin{bmatrix} x_l \\ x_{r_1} \\ x_{r_2} \\ \vdots \\ x_{r_n} \end{bmatrix}, \tag{6.1}$$

where $\widehat{X}_I$ is the aggregate vector of IPU poses from the previous chapter. The time-update equations of the Kalman filter remain unaltered, effectively assuming, as was the case for projectors, that the state of the display surface remains mostly constant, but may drift over time or occasionally undergo small changes. Of course, a different motion model could be used for the display surface state depending on the specific situation.

These changes to the state vector require a corresponding update to the filter's error covariance matrix previously defined in Equation 5.2:

$$\widehat{P}_k = \begin{bmatrix} \widehat{P}_I & P_{IS} \\ P_{SI} & P_S \end{bmatrix}, \quad \widehat{P}_I = \begin{bmatrix} P_l & P_{r_1,l} & P_{r_2,l} & \dots & P_{r_n,l} \\ P_{l,r_1} & P_{r_1} & 0 & \dots & 0 \\ P_{l,r_2} & 0 & P_{r_2} & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ P_{l,r_n} & 0 & \dots & 0 & P_{r_n} \end{bmatrix}, \tag{6.2}$$

where P_S is the error covariance of the display surface state, $\widehat{P}_I$ is the matrix of IPU pose covariances from the previous chapter, and $P_{IS} = P_{SI}^T$ is the cross-covariance between the IPU poses and the display surface state. Equation 5.5 is correspondingly updated to include process noise for the display surface

state vector Q_S:

$$\widehat{Q}_k = \begin{bmatrix} \widehat{Q}_I & 0 \\ 0 & Q_S \end{bmatrix}, \tag{6.3}$$

where $\widehat{Q}_I$ is the process noise of the IPU poses from before, and Q_S is the process noise of the display surface state.

Finally, the Jacobian matrix $\widehat{H}_k$ must be updated to include the sensitivity of the measurements to changes in the display surface state as follows

$$\widehat{H}_k = \begin{bmatrix} H_l & 0 & 0 & \dots & 0 & H_{l,S} \\ H_{l,r_1} & H_{r_1,l} & 0 & \dots & 0 & H_{r_1,S} \\ H_{l,r_2} & 0 & H_{r_2,l} & & \vdots & H_{r_2,S} \\ \vdots & \vdots & & \ddots & 0 & \vdots \\ H_{l,r_n} & 0 & \dots & 0 & H_{r_n,l} & H_{r_n,S} \end{bmatrix}, \tag{6.4}$$

where

$$H_{l,S} = \frac{\partial h_l(x_l^-; z_l^s ...)}{\partial x_s} \tag{6.5}$$

$$H_{r_i,S} = \frac{\partial h_{r_i}(x_l^-, x_{r_i}^-; z_{r_i,l}^p, \dots)}{\partial x_s}. \tag{6.6}$$

While the above is the extent of the mathematical changes that must be made to the filter in order to continuously estimate changes in the desired display surface parameters, there are additional issues related to observability and maintaining global consistency of the display surface state that must be dealt with in practice. These issues are discussed in Sections 6.2 and 6.3.

6.2 Maintaining Global Consistency

The state of the display surface can be thought of as a set of global state parameters that all IPUs may alter in the course of performing their filter updates. It is thus important to ensure that the state of the

display surface remains globally consistent as it is updated by each of the IPUs.

In the current implementation, global consistency of the display surface is maintained through the use of a token passing approach, where the IPUs perform their filter updates sequentially in a round-robin fashion. When an IPU receives the token, it sends a request to the other IPUs in the display for their matching camera image and current pose estimate as described in the previous chapter. After receiving a response from the other IPUs, it performs its Kalman filter update and distributes the updated state of the display surface to the other IPUs before passing on the token. This ensures a globally consistent estimate of the display surface at the cost of serializing the operation of the IPUs.

This need to serialize the operation of the IPUs when simultaneously estimating both the poses of the IPUs and the display surface can, however, be eliminated in future work via incorporating the work of [Rao and Durrant-Whyte, 1991], which describes a method of decentralizing a Kalman filter among a number of independent sensor nodes. The authors show that even though each node processes only its own local measurements and updates its own local copy of the state and error covariance, it is possible to achieve a globally consistent estimate of the state and error covariance by having each sensor node broadcast its local estimates of the *a priori* and *a posteriori* state and error covariance to all other sensor nodes. All nodes then incorporate this remote information into their own local estimates using the so-called "assimilation" equations, which guarantee not only that the resulting estimate is consistent among all nodes, but also that it is *equivalent* to that which would have been computed by a single global Kalman filter processing all measurements in a centralized fashion.

As suggested by Greg Welch, accurately reflecting any constraints provided by the surface shape can also be beneficial in achieving a globally consistent solution. For example, if it is known that the display surface is a cylinder of unknown radius, using a parametric cylinder representation for the surface, e.g. radius, height, position, and orientation, enforces the cylindrical constraint. As various IPUs update the state of the display surface, its shape is guaranteed to remain cylindrical.

6.3 Observability

Recall from Chapter 1 that in order for geometric image correction to be successful, the calibration of all projectors, the geometry of the display surface, and the position of the viewer must be known

with respect to some coordinate system or *frame of reference*. In displays where only a subset of the above parameters are estimated, it is important that this frame of reference be maintained so that the remaining parameters continue to be meaningful. For example, in the extensions proposed in this chapter, parameters of the display surface and the poses of all projectors are estimated continuously, but the position of the viewer is not. The estimation algorithm must therefore ensure that estimates are linked to the original frame of reference lest knowledge of the viewing location relative to the projectors and display surface is lost.

In general, this becomes a question of whether the state parameters that are being estimated are observable in the original coordinate system, which can be verified by testing the rank of the observability matrix as described in Section 3.4.1. In the previous chapter, the entire display surface was assumed to be known and static, allowing it to act as the known frame of reference, in a sense "anchoring" the pose estimates of the IPUs to the original coordinate system. A link to the original coordinate system can also be maintained in other ways. For example, this role could be fulfilled by an IPU whose pose is static, fiducials placed in a static area of the surface, or some triangle of the display surface mesh that is assumed to be static and is not included in the filter state vector.

In some displays, it may only be important that the state parameters are observable *relative* to one another rather than in the original coordinate frame. For example, in the case of a dynamic shader lamps display, there is no concept of a viewing location. The only concern is that the projected imagery be properly registered to the surface, which requires only that the poses of the IPUs be correct *relative* to the display surface or vice versa. Display parameters can be estimated relative to the display surface or the pose of one IPU simply by not including its pose in the filter state vector. This effectively makes it the known coordinate system with respect to which the other state parameters are estimated. In practice, this may be necessary to prevent the system from being inherently unobservable due to the fact that any 6-DOF transformation applied to both the surface and all projectors will have no effect on the measurement residuals. This unobservability could result in a tendency for the poses of the IPUs and surface to drift over time, potentially causing numerical instability.

6.4 Detecting Motion

In the distributed cooperative framework of the previous chapter, motion of the IPUs could only be detected indirectly via examining changes in the poses of the IPUs as driven by the measurement residuals. One consequence of this is that motion of one IPU may result in undesired changes to the poses of other stationary IPUs or the display surface state. Consider a two-IPU display where one IPU is moved while the other remains stationary. In this situation, the motion of one IPU will cause both IPUs to encounter increased measurement residuals due to the processing of remote correspondences. Since no information about which IPU is moving is provided to the filter, significant changes may be made to the stationary IPU's pose during this time. Though these changes may later be reversed as the filter accumulates more information about the system state, changes in the imagery of stationary projectors can be visually distracting.

The underlying problem is that, using only reprojection errors between cameras, it is not possible to determine in all cases *which* of the IPUs and/or display surface is undergoing changes. Indeed, the $\widehat{Z}_k - \widetilde{Z}_k$ term in Equation 5.7, often called the *measurement residual* or *measurement innovation* is a vector of reprojection errors. Since reprojection errors depend only on the *relative* placement and shape of the display surface and IPUs, it may be ambiguous whether the residuals arise due to errors in the poses of one or more projectors, errors in the state of the display surface model, or some combination of both.

As an example of the ambiguity associated with reprojection error, consider a single IPU display that uses only local correspondences. In terms of reprojection error, there is no difference between translating the entire display surface x units to the left and translating the IPU x units to the right. After performing either transformation, the IPU is at the same location *relative* to the display surface. It is thus impossible to determine in this case whether the IPU has moved or whether the surface has moved.

One way this ambiguity can be resolved is by adding sensors to the IPUs that give a direct indication of motion. The current implementation of the system makes use of inertial motion sensors attached to each IPU to query at high rates whether an IPU is moving at any instant in time. When

motion of an IPU is detected, the amount of process noise, or uncertainty, in its pose estimate is substantially increased via the process noise matrix $\widehat{Q}_k$ (Equation 5.5), which describes how uncertainty in the various estimated parameters increases as a function of time between filter updates. This indicates to the filter that it should be *less* certain of the pose of this IPU relative to the other IPUs in the display and allows it to make the appropriate corrections to the state.

6.5 Application Specific Implementations

In this section, two extensions to the framework of the previous chapter are examined that estimate information about the display surface under different surface representations and assumptions. A dynamic shader lamps scenario is first considered where the display surface is rigid with known geometry, but may be moving over time. This is followed by a description of how the framework can be used to refine the vertices of a static display surface represented by a polygonal mesh in order to remove small errors.

6.5.1 Dynamic Shader Lamps

This situation was described briefly earlier. We have a display surface that is rigid in relation to itself, but the entire surface may move over time. In this case, the basic geometry of the surface is unchanging and can be considered an intrinsic parameter of the surface that need not be part of the display surface state vector x_s. The extrinsics of the display surface must, however, be estimated. In general, this consists of a 6-DOF pose that is conceptually the same as the pose of a projector

$$x_s = \begin{bmatrix} S_x & S_y & S_z & S_\psi & S_\theta & S_\phi \end{bmatrix}^T, \tag{6.7}$$

where (S_x, S_y, S_z) is the location of the display surface and $(S_\psi, S_\theta, S_\phi)$ is its orientation.

Since it is known that the display surface is or could be moving, the process noise of the display surface pose is estimated dynamically, such that when the surface is moving, a larger amount of process noise, or uncertainty, is added to its pose relative to when it remains stationary. This indicates to the filter that it should be less certain of the pose of the display surface relative to the poses of any IPUs that may be stationary at the time (which will have a lower, constant amount of process noise).

Figure 6.2: A model of a house with texture provided by two projectors. The pose of the house is estimated continuously, allowing the projected imagery to remain registered to the house as it is moved to different orientations and positions by the user.

This dynamic shader lamps extension has been demonstrated in a two-IPU display consisting of a textureless doll house that is augmented with projected imagery, as seen in Figure 6.2. The pose of the house is estimated continuously using the cooperative continuous calibration framework such that the projected imagery remains registered to the house as it is moved by the user to various orientations and positions. As the house remains stationary, the IPUs can also be moved and automatically recalibrated.

6.5.2 Polygonal Mesh Refinement

This section describes how the distributed cooperative framework can be used to improve geometric correction results in a multi-projector display by removing small, local errors in the vertices of a static display surface represented as a polygonal mesh.

The display is illustrated in Figure 6.3 and contains two projectors that illuminate a display surface formed by three walls of a room that intersect in two corners. Initial calibration of the display was performed upfront using structured light projection with a process that reconstructs each wall of the surface as an infinite plane and intersects these planes to form a polygonal model of the surface. Unfortunately, due to slight deviations from planarity in each of the walls, this initial calibration resulted in significant errors in the corner regions as seen in Figure 6.4a and b, which shows imagery projected into the corner with and without the underling mesh representation superimposed. Due to the sharp discontinuity in the surface, errors in this area are likely to be disturbing to viewers.

In order to remove these errors using the cooperative calibration framework, each wall's planar

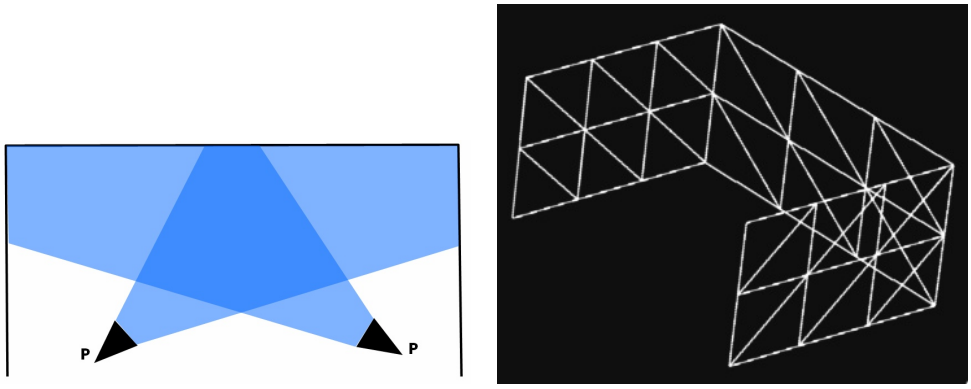


Figure 6.3: Left: The layout of a two-projector display as seen from above with a display surface consisting of two simple room corners. Right: The internal representation of the surface as a polygonal mesh. Due to deviations from planarity, each wall is subdivided into a small number of triangles.

representation is first divided into a number of smaller facets to provide the freedom in the surface representation necessary to achieve accurate corner registration. The internal mesh representation of the surface is depicted in Figure 6.3.

Since the errors in the display surface mesh are small, only the vertices of the mesh are included in the display surface state x_s, and it is assumed that the connectivity information does not change. However, by re-evaluating the connectivity of the vertices after each update to the mesh, it should be possible to accommodate larger changes to the surface. The display surface state vector estimated by the filter is thus

$$x_s = \begin{bmatrix} v_1 & v_2 & \dots & v_\eta \end{bmatrix}^T, \tag{6.8}$$

where each v_i is a 3-vector representing a vertex of the mesh. Since the surface is assumed to be static, a small amount of constant process noise is added to the display surface state vector to allow the filter to correct the errors in the display surface vertices.

Figure 6.4c shows the results of running the continuous calibration algorithm with polygonal mesh refinement on the surface after only a few seconds. The corner vertices in the display surface model are now accurately registered to the true room corner, allowing accurate geometric image correction. The discontinuity in the projected imagery as it crosses the corner has now been eliminated.

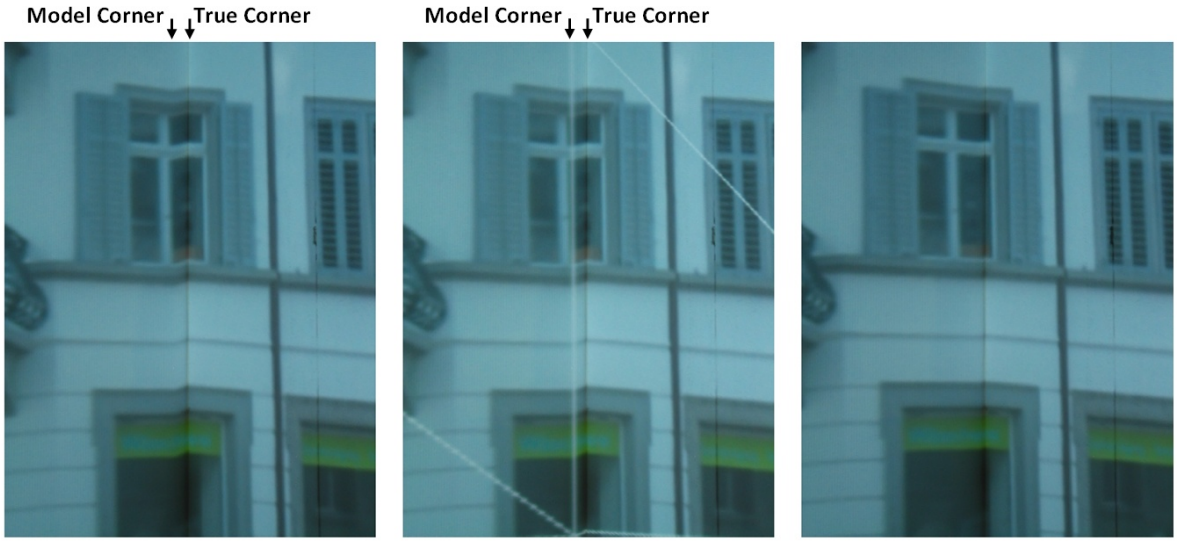


Figure 6.4: Room corner registration before continuous calibration with and without display of underlying mesh (a & b). Room corner registration improves noticeably after a few seconds of continuous calibration (c).

Global Consistency

Since the initial errors in the display surface estimate are small, it was not necessary in this example to enforce global consistency in the display surface model between IPUs by communicating changes in the vertices between IPUs. While this may be acceptable for displays where the poses of the IPUs are not expected to change significantly, there may be scenarios where global communication of display surface updates are required. In this case, it is possible to employ the methods for maintaining global consistency described in Section 6.2.

Surface Representation

While it would be desirable to represent the display surface as a number of infinite planes instead of a polygonal mesh as described in Section 6.1.1, ultimately it was not practical since many of the planes would have been nearly coplanar, leading to issues when intersecting the planes to form a polygonal model of the surface for performing two-pass rendering.

Performance

When used on a large mesh with many vertices, computational performance is likely to be poor due to the large size of the state vector. Recall from discussion in the previous chapter that the complexity of the measurement update step is $O(m^3+n^3)$ for batch processing or $O(mn^3)$ for sequential processing with m the number of measurements to be processed and n the size of the state vector. However, if we can assume the vertices of the mesh are statistically independent, a sequential measurement processing approach suddenly becomes much more attractive since, as each measurement is processed, the Kalman filter state vector need only include the locations of at most three vertices in addition to the projector poses. This is because assumed independence of the vertices means that the processing of a measurement requires only that we update the vertices of the one face in the mesh from which the measurement originated.

In a sequential measurement processing approach, the form of the sequential Kalman filter state vector $\widehat{X}_s$, error covariance $\widehat{P}_s$, and measurement Jacobian H_s would be

$$\widehat{X}_s = \begin{bmatrix} \widehat{X}_I \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad \widehat{P}_s = \begin{bmatrix} \widehat{P}_I & 0 \\ 0 & \widehat{P}_v \end{bmatrix}, \quad H_s = \begin{bmatrix} H_{l,r_i} & 0_{2,6(i-1)} & H_{r_i,l} & 0_{2,6(n-i)} & H_{v_1} & H_{v_2} & H_{v_3} \end{bmatrix}, \tag{6.9}$$

where independence between the vertices and the poses of the IPUs is assumed, and $0_{m,n}$ denotes an m x n matrix of zeros, H_{v_i} is the Jacobian of the measurement function with respect to vertex v_i, and $\widehat{P}_v$ is the aggregate matrix of individual vertex covariances:

$$\widehat{P}_v = \begin{bmatrix} P_{v_1} & 0 & 0 \\ 0 & P_{v_2} & 0 \\ 0 & 0 & P_{v_3} \end{bmatrix}. \tag{6.10}$$

In this formulation, the computational complexity for large meshes is greatly reduced since the cost of performing the filter update is independent of the size of the mesh. Determining the face of

the mesh on which a measurement lies is, however, dependent on the size of the mesh and efficient techniques for ray-casting should be employed as discussed in the previous chapter in Section 5.5.3.

6.6 Summary

This chapter has described how a distributed cooperative approach can be used to estimate the poses of all projectors as well as information about the display surface in a unified fashion. Actual results were demonstrated using two practical display scenarios: a dynamic shader lamps display where the display surface position and orientation is estimated continuously and a display where continuous calibration is used to refine the vertices of a display surface represented as a polygonal mesh.

Beyond the display scenarios presented here, it should be possible to use this distributed cooperative framework in additional applications of continuous calibration. For example, as projectors are repositioned to illuminate portions of the display surface that do not yet exist in the display surface model, one can imagine extending the surface by initializing new vertices or new sets of parameters that are subsequently refined using cooperative continuous calibration. A variety of additional future extensions are described in Chapter 7.

CHAPTER 7

SUMMARY AND FUTURE WORK

The previous chapters have directly supported the central thesis of this dissertation:

Geometric calibration of a multi-projector display can be maintained continuously, during system operation, using a distributed cooperative approach that simultaneously refines

I. *the poses of multiple projectors and*

II. *the geometry of the display surface.*

Chapter 5 introduced a novel cooperative calibration framework for multi-projector displays and described how it could be used to estimate the poses of all projectors in a multi-projector display. This framework was built upon the concept of using groups of intelligent projector units (IPUs), projectors combined with cameras and computation, that exchange information across a network to cooperate in estimating the poses of all projectors. This was made possible through the collection of both local correspondences (obtained by each IPU independently) and remote correspondences (obtained via communication with other IPUs). A Kalman filter was used at each IPU to process these local and remote correspondences into an estimate of its pose that is subsequently communicated to the other IPUs in the display. The details of how this framework can be implemented are also described, and practical results from an actual two-projector display with multi-planar display surface were provided, although the implementation has been demonstrated with as many as three IPUs.

Chapter 6 described how this framework can also be used to estimate the poses of all projectors in a multi-projector display as well as information about the display surface in a unified fashion, resulting in a complete framework for continuous geometric calibration in a multi-projector display. Actual results from two practical scenarios were provided. The first described how the pose of a rigid

display surface with known geometry can be estimated in addition to the poses of the projectors for use in a dynamic shader lamps [Bandyopadhyay et al., 2001] scenario. In the second, the cooperative calibration framework was used to eliminate errors in the vertices of a display surface represented as a polygonal mesh.

7.1 Future Work

There are many additional research directions that could be explored beyond the work presented in this dissertation. A few of these are described here.

- It would be interesting and worthwhile to explore using the distributed cooperative framework to estimate additional geometric parameters of the display. For example, as projectors are repositioned by the user, it may be necessary to change the focus of the projector to maintain image quality. This could be accommodated by continuously estimating the intrinsics of the projector in addition to its pose. Also, the repositioning of a projector may cause new portions of the display surface to be illuminated that do not yet exist in the display surface model. In this case, one can imagine extending the surface by initializing new vertices or new sets of parameters that are subsequently refined using cooperative continuous calibration.

- Beyond geometric calibration, the framework could be extended to perform continuous *photometric* calibration, for example continuous estimation of blending masks or continuous luminance and color correction. This cooperative framework may also be useful in estimating photometric properties of the display surface such as its bidirectional reflectance distribution function (BRDF), by cooperatively integrating photometric measurements of the surface captured from various perspectives by multiple IPUs.

- One area where this framework could be of benefit is projectors in office environments. As suggested by Greg Welch, continuous calibration could be used in this environment to maintain a consistent (and measurable) display quality over time, possibly alerting the user to the fact that display contrast has decreased to the point where a projector bulb needs to be replaced.

- The framework could also be improved through the incorporation of additional sensors, such

as an off-the-shelf orientation sensor, that can be queried at high rates to augment the vision-based pose estimation. As suggested by Greg Welch, one possible way of achieving this is through the use of a *complementary* or *error-state* Kalman filter [Maybeck, 1979] as illustrated in Figure 7.1. In this design, estimates from the additional sensor, such as an orientation sensor in the figure, can be used as soon as they are available and are corrected using an error term produced by a Kalman filter. This filter estimates the difference between the values produced by the sensor and vision systems, in effect estimating the error in the sensor output using the vision system output as a reference. This allows the vision system to correct for any long term drift or bias in the sensor estimates, while allowing the use of low latency updates from the sensor itself. This can be implemented in either a feed-backward or feed-forward configuration as shown in Figure 7.1. Typically, a feed-backward approach is desirable due to the potential for error estimates in the feed-forward implementation to grow unbounded as the sensor estimate drifts over time.

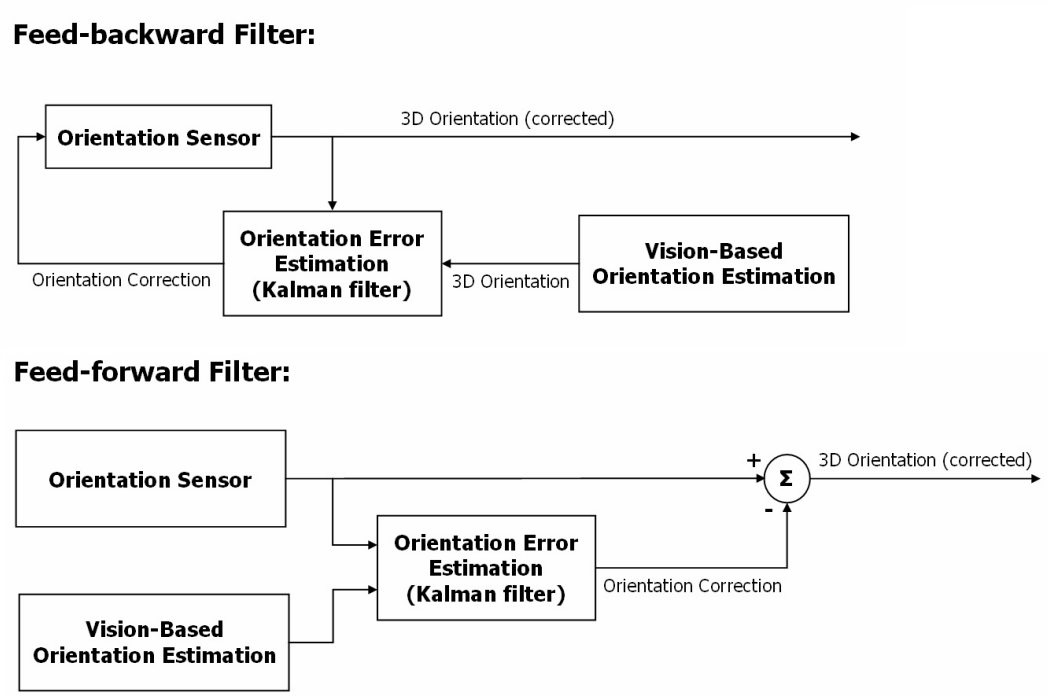


Figure 7.1: Error-state Kalman filter for incorporating additional sensors (orientation sensor in this example) in feed-forward and feed-backward configurations.

APPENDIX A

ADDITIONAL CONTRIBUTIONS

This appendix contains additional published contributions whose contents have not been included in the body of the dissertation. The first work examines continuous projector pose estimation in a single projector display when cameras are not mounted to the projector itself, but rather to a separate rigid structure in the vicinityof the display. The second work describes how two-pass rendering can be used to create a unifying framework for many tasks in projector-enhanced office environments. These tasks include displaying on the surface of an office cubicle, camera-based scanning of documents, and the use of focus-plus-context displays [Baudisch et al., 2001].

Real-Time Projector Tracking on Complex Geometry Using Ordinary Imagery

Tyler Johnson and Henry Fuchs
Department of Computer Science
University of North Carolina Chapel Hill

Abstract

Calibration techniques for projector-based displays typically require that the display configuration remain fixed, since they are unable to adapt to changes such as the movement of a projector. In this paper, we present a technique that is able to automatically recalibrate a projector in real time without interrupting the display of user imagery. In contrast to previous techniques, our approach can be used on surfaces of complex geometry without requiring the quality of the projected imagery to be degraded. By matching features between the projector and a stationary camera, we obtain a new pose estimate for the projector during each frame. Since matching features between a projector and camera can be difficult due to the nature of the images, we obtain these correspondences indirectly by first matching between the camera and an image rendered to predict what the camera will capture.

Figure 1. Our tracking process adapts to changes in projector pose, allowing "on the fly" display reconfiguration.

1. Introduction

Research in adaptive projector displays has enabled projection on display surfaces previously thought impractical. Raskar [16] and Bimber [3] describe general methods of correcting for the geometric distortions that occur when non-planar surfaces are used for display. Other work [2, 10, 13] has focused on eliminating the need for high-quality projection screens by compensating for the color and texture of the display surface. These techniques and others have greatly increased the versatility of the projector and brought us closer to a "project anywhere" display.

Recently, the focus has begun to shift towards robust automatic calibration methods [1, 15, 17] that require little or no interaction on the part of the user. Within this category are techniques that can perform calibration while user imagery is being projected. This allows display interruption to be avoided in the event that the display configuration changes, *e.g.* a projector is moved.

These "online" auto-calibration techniques can be divided into two categories. In the first are the active techniques where calibration aids that are imperceptible to a human observer are injected into the user imagery. Cotting et al.[5, 6] embed these imperceptible calibration patterns in the projected imagery by taking advantage of the micromirror flip sequences used to form images in DLP projectors. Image intensity is modified slightly at each pixel so that a camera exposed at a small frame interval will capture the desired calibration pattern. This technique currently requires that a portion of the projector's dynamic range be sacrificed, leading to a slight degredation of the user imagery.

The second type of auto-calibration technique does not rely on the use of calibration aids and instead attempts to extract calibration information from the user-projected imagery itself. In Yang and Welch [20], features in the user imagery are matched between a pre-calibrated projector and camera and used to automatically estimate the geometry of the display surface. This technique leaves the user imagery unmodified, but relies on the presence of features in the user imagery that are suitable for matching.

To our knowledge, no existing work has demonstrated the ability to continuously track a projector in real-time on complex display surface geometry without modifying the projected imagery or using fixed fiducials. In addition to

allowing dynamic repositioning of the projector during display, such a system could also be used for projector-based augmentation. Here, a hand-held projector can be used to augment objects with information or to alter their appearance [17, 18]. This effectively allows a projector to be used as a graphical "flashlight" where the projector can be controlled directly by the user to create imagery on previously unlit portions of a surface or to enhance image detail by moving the projector closer to the surface.

In this paper, we describe a technique that enables such a system. By matching features between the projector and a stationary camera, we re-estimate the pose of the projector continuously. Since matching features between a projector and camera directly can be difficult, we propose obtaining these correspondences indirectly by first matching between the camera and an image generated to predict what the camera will capture. We present a simple radiometric model for generating this predicted image. Our technique does not affect the quality of the projected imagery and can be performed continuously without interrupting the display.

2. Predictive Rendering for Feature Matching

Given a static camera of known calibration, where the depth at each camera pixel is also known, a set of image correspondences between the camera and a projector indirectly provides a set of 2D-3D correspondences in the projector, allowing the calibration of the projector to be determined. Unfortunately, there are a variety of factors that complicate the process of matching between projector and camera images as seen in Figure 4. Some of these complications include

1. differences in resolution, aspect ratio and bit-depth
2. large camera-projector baselines
3. radiometric effects *e.g.* projector/camera transfer function and surface BRDF present in camera image, but not in projector image

To avoid these problems, we instead perform matching between an actual image captured by the camera and a prediction of this captured image generated using graphics hardware. We will refer to the image sent to the projector as the *projected* image, the image captured by the camera as the *captured* image, and the prediction of the captured image as the *predicted* image.

Since the displacement of a projector within a single frame is small, geometric differences between the predicted and captured images will be small as well. Also, by generating a predicted image that takes into account the radiometric properties of the display, the captured and predicted images will be similar in intensity, leading to better matching results. By reversing the process used to generate the predicted image, we can map features in the predicted image to their original locations in the projected image to obtain 2D-3D correspondences allowing us to calibrate the projector.

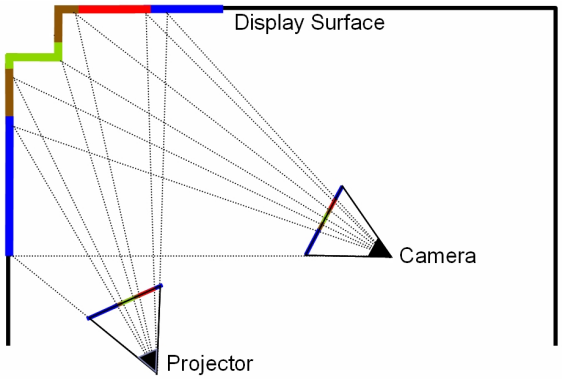


Figure 2. Mapping projector pixels to camera pixels.

2.1. Geometric Prediction

The first step in generating the predicted image is to determine the mapping from projector pixels to camera pixels. This allows us to warp the projected image into the frame of the camera as if it had observed the projected imagery. This mapping is completely defined by the calibration of the projector and camera and the geometry of the display surface and is the same process needed to determine the required warping to compensate for the display surface geometry when displaying a desired image to a viewer. In the case of generating the predicted image, we map between projector and the camera instead of between the projector and the viewer.

We accomplish the warping between the camera and projector using the second pass of the two-pass rendering technique proposed by Raskar [16]. Using the projection matrix of the projector, we project the texture coordinates of the projected image onto the display surface geometry and render it using the projection matrix of the camera. An illustration of this process is provided in Figure 2.

2.2. Radiometric Prediction

As Tables 1 and 2 show, incorporating a radiometric simulation into the generation of the predicted image can result in an immense improvement in feature matching performance, especially for rendered imagery. The radiometric model we use assumes that the projector can be reasonably approximated as a point light source, which implies that each point on the display surface receives illumination from only one direction. We will additionally assume a display surface with a uniform, diffuse BRDF since this type of surface is commonly used for projective display. We also ignore any contribution to the captured image from indirect

illumination.

For a point X on the display surface, we will denote its projection in the camera and projector images as x_c and x_p. Given one of the these points, the locations of the other two are easily determined using the calibration of the camera and projector and the model of the display surface.

The radiometric value measured by a camera sensor is irradiance, the amount of energy per area. The irradiance within a sensor pixel is integrated over time to produce the sensor's output. A pixel's intensity value in the final image may however be a non-linear function of the sensor output. The goal of our radiometric simulation is then to predict the irradiance arriving at each camera pixel and apply the transfer function of the camera, also called its input-output response function or simply response. Let R_c be the response function of the camera and $E(x_c)$ the irradiance at a camera pixel x_c. The intensity in the captured camera image $M_c(x_c)$ is then

$$M_c(x_c) = R_c(E(x_c)). \tag{1}$$

The irradiance at a point on the camera sensor is a function of the scene radiance, the energy per area per solid angle of the source. Assuming a small aperture and a thin lens, the irradiance $E(x)$ at a point x on the camera sensor is directly proportional to the scene radiance L arriving at x [9]. Specifically, this proportionality is

$$E(x) = L\frac{\pi}{4}\frac{cos^4\theta}{n^2}, \tag{2}$$

where n is the f-number of the lens and θ is the angle between the normal of the sensor plane and the unit vector from x to the center of the lens. This equation indicates a variation in the proportionality between scene radiance and sensor irradiance over the camera's field-of-view. We have found this variation to be negligible and instead use the simplification

$$E(x_c) = \alpha L\left(X, \overrightarrow{XC}\right), \tag{3}$$

where $L\left(X, \overrightarrow{XC}\right)$ is the radiance at X in the direction of the camera center C, and α is some constant.

The value of $L\left(X, \overrightarrow{XC}\right)$ is the result of illumination from the projector being reflected by the display surface towards the camera and depends on the surface BRDF. Since we consider the projector to be a point light source, the point X receives illumination only along the direction from the projector center $\overrightarrow{XP}$. This, combined with the assumption of a uniform, diffuse surface, leads to

$$L\left(X, \overrightarrow{XC}\right) = \rho E\left(X, \overrightarrow{XP}\right), \tag{4}$$

where $E\left(X, \overrightarrow{XP}\right)$ is the incident irradiance at X due to the projector and ρ is the surface BRDF.

The incident irradiance at a surface point due to a point light source depends on the radiant intensity of the light source in the direction of the surface point, the orientation of the surface normal with respect to the incoming light direction, and the distance to the light source. If $I\left(P, \overrightarrow{PX}\right)$ is the radiant intensity of the projector in the direction of point X, then

$$E\left(X, \overrightarrow{XP}\right) = I\left(P, \overrightarrow{PX}\right)\frac{cos(\theta)}{r^2}, \tag{5}$$

where θ is the angle between the surface normal at X and the direction $\overrightarrow{XP}$ and r the distance between X and the projector.

We model the radiant intensity $I(P, \overrightarrow{PX})$ as a function of the intensity of the projected image at pixel x_p, $M_p(x_p) = [r,g,b] \in [0,1]^3$, and the response function R_p of the projector such that

$$I\left(P, \overrightarrow{PX}\right) = S(x_p)\left[R_p(M_p(x_p)) \cdot [I_r, I_g, I_b]\right], \tag{6}$$

where $I_{r..b}$ are the maximum radiant intensities of the red, green and blue color channels of the projector and $S \in [0,1]$ is a per-pixel attenuation map. The purpose of S, which we refer to as the projector *intensity profile*, is to model changes in brightness over a projector's field-of-view due to effects such as vignetting.

Combining the above equations, the final intensity value measured at each camera pixel as a function of the intensity at its corresponding location in the projected image is

$$M_c(x_c) = R_c\left(\frac{cos(\theta)}{r^2}S(x_p)\left[R_p(M_p(x_p)) \cdot \left[\bar{I}_r, \bar{I}_g, \bar{I}_b\right]\right]\right), \tag{7}$$

where we have combined α, ρ and the $I_{r..b}$ into the terms $\bar{I}_{r..b}$. Since the field-of-view of each camera pixel may encompass more than a single projector pixel, we filter the projected image to obtain an average intensity value that we use to evaluate Equation (7).

3. Calibration

In this section, we describe our process of calibrating the geometric and radiometric parameters necessary to apply the technique we use to generate the predicted image.

3.1. Geometric Calibration

We accomplish initial geometric calibration in an up-front step by observing projected structured light patterns

with a stereo camera pair. This process yields a set of correspondences between the projector and camera pair that allows us to reconstruct the geometry of the display surface and calibrate the projector. Raskar provides a thorough discussion of this process in [16]. To obtain an accurate model of our room-like display surface as seen in Figure 2, we use the RANSAC-based plane fitting technique introduced by Quirk [14]. Any other reconstruction method that gives a geometric description of the surface could also be used.

3.2. Radiometric Calibration

In addition to the geometric calibration, generation of the predicted image requires estimation of the projector and camera response, the projector intensity profile, and the terms $\bar{I}_{r..b}$. Since these are intrinsic properties, they can be calibrated once in an up-front process and then used in arbitrary geometric configurations.

Since we use the stereo camera pair for geometric calibration, out of convenience, we also use one of these cameras for projector tracking. To prevent indirect scattering from affecting the results, we perform radiometric calibration by projecting on a portion of the display surface that produces low levels of these effects. In our case, we project onto a single plane and use our geometric calibration process to establish the geometric relationship between the camera, projector and display surface.

3.2.1 Projector Response

The first step in our radiometric calibration process is to calibrate the projector response. We accomplish this manually by projecting an image where half the image is filled with some intensity I and the other half with a dither pattern computed using error-diffusion dithering [8] to have a certain proportion of black and full intensity pixels. By adjusting the proportion of black and white pixels such that the intensity of the image appears consistent throughout, we can determine what proportion of the projector's maximum output intensity the intensity I represents. This is done for a number of intensities and the resulting data points are interpolated with a spline to reconstruct the response function of the projector.

3.2.2 Intensity Profile

We recover the intensity profile of the projector by projecting a sequence of solid grayscale images of increasing intensity. To remove the effect of projector response on the captured camera images, we linearize the output of the projector during this step using the inverse of the projector response.

At each pixel in a captured camera image, we have

$$\upsilon(x_c) = \frac{r^2 R_c^{-1}(M_c(x_c))}{cos(\theta)} = S(x_p)[M_p(x_p) \cdot [\bar{I}_r, \bar{I}_g, \bar{I}_b]]. \quad (8)$$

The value of υ at each camera pixel can be evalutated given the response function of the camera. Since we have linearized the projector response when projecting the sequence of grayscale images, we can use the captured camera images to get a rough estimate of the camera response. We do this by finding the median intensity value in each camera image and use this as the camera response for the grayscale intensity of the projected image. In the next section, we will describe our technique for refining the camera response estimate.

Next, we compute $\upsilon(x_c)$ at each pixel of the camera images. Let x_m be the pixel in an image where the maximum value of υ occurs. Dividing the value of υ at each pixel by $\upsilon(x_m)$ gives the relation

$$\frac{S(x_p)}{S(x_{p_m})} = \frac{cos(\theta_m) r^2 R_c^{-1}(M_c(x_c))}{cos(\theta) r_m^2 R_c^{-1}(M_c(x_m))}, \quad (9)$$

with subscript m indicating values for pixel x_m. The $M_p(x_p) \cdot [\bar{I}_r, \bar{I}_g, \bar{I}_b]$ terms in both the numerator and denominator cancel since the projected image was uniform in intensity. Because $\upsilon(x_m)$ is a maximum in the camera image, $S(x_{p_m}) = 1$, and we can compute an estimate of the intensity profile of the projector at each pixel using a single camera image.

Since the computed intensity profile may vary between images, we use the average of the intensity profiles extracted from each image as our final estimate. It is important in this step to exclude camera images containing saturated pixels, which will affect the computation of the intensity profile. We also blur the extracted profile with a small gaussian kernel to remove noise.

3.2.3 Projector Brightness and Camera Response

The remaining terms to be calibrated are the $\bar{I}_{r..b}$ and the refinement of the camera response. The process we use for calibrating the camera response is based on the technique described in Debevec [7].

In addition to the grayscale images used to calibrate the intensity profile of the projector, we also project a series of solid color images of different intensities, linearizing the output of the projector as we did before. From Equation (7), at each pixel in a camera image taken of one of these solid color images, we have a linear equation in the $\bar{I}_{r..b}$ and one of the 256 discrete values that form the domain of R^{-1}. If the projected intensity is (r, g, b) and the intensity measured by the camera at some pixel is i, we have

$$sr\bar{I}_r + sg\bar{I}_g + sb\bar{I}_b - R_c^{-1}(i) = 0, \quad (10)$$

where $s = \frac{cos(\theta)}{r^2} S(x_p)$.

Let $g_i = R^{-1}(i)$ and let z_k be a vector of length 256 composed of all zeros except that it contains the value 1 at the location corresponding to the intensity value measured by the camera in the kth equation. The system of equations can then be put into matrix form as

$$\begin{bmatrix} s_1 r_1 & s_1 g_1 & s_1 b_1 & z_1 \\ s_2 r_2 & s_2 g_2 & s_2 b_2 & z_2 \\ & \ddots & & \\ s_n r_n & s_n g_n & s_n b_n & z_n \end{bmatrix} \begin{bmatrix} \bar{I}_r \\ \bar{I}_g \\ \bar{I}_b \\ g_1 \\ \vdots \\ g_{256} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (11)$$

This system can be efficiently solved using SVD to produce the best solution vector in the least-squares sense. In practice, it may be possible that no camera pixel of a certain intensity can be found in any of the camera images. In this case, the values of R_c^{-1} for which data exists can be calibrated and then interpolated to produce the missing values. To enforce smoothness between the g_is, we also add a second derivative term to Equation (11) of the form $\lambda(g_{i-1} - 2g_i + g_{i+1})$ where λ can be used to control the smoothness.

Since the solution vector is only defined up to an arbitrary scale and sign, we choose the sign such that the solution vector is positive and the scale such that the g_i lie in the range $[0,1]$. The response function of the camera is easily obtained by inverting the g_is.

In our testing, we captured images of 17 different intensities each of red, green, blue, and yellow. In choosing the camera image pixels we use as contraints in Equation (11), we have created an automated process that selects pixels uniformly across color, intensity, and image location.

3.2.4 Error

To estimate the error in our calibration and validate our model, we used the calibration to predict the camera images used in calibrating the projector intensity profile. Comparing the predicted images to the actual camera images, we found the overall average error to be just under 4 camera intensity values.

4. Rendering

We implemented our radiometric model in a real-time GPU pixel shader that allows predicted images to be produced at interactive rates. The shader takes as input all of the radiometric parameters with the projector and camera response stored as 1D textures and the intensity profile stored as a 2D texture. The geometric parameters are two 2D floating-point textures called the geometry and normal map, which store the (x,y,z) position and normal of the display surface at each camera pixel. The projection matrix of the projector and its center-of-projection are also input parameters.

To predict the intensity measured by the camera at each pixel, the shader program first estimates the average color of the projected image pixels that fall within the extent of each camera pixel. At each predicted image pixel, we look up the display surface vertex X in the geometry map and project it into the projected image using the projector calibration to obtain a texture coordinate. We repeat this process at three neighboring pixels, allowing us to compute two derivative vectors indicating the size of the region in the projected image that should be filtered. We pass this information to the texture mapping hardware, which filters the projected image and the intensity profile using a kernel of the appropriate size. We next apply the projector response to the filtered color by performing a per channel look-up in the projector response texture.

Using the center-of-projection of the projector and the value of X, the value of r^2 for the pixel is easily computed. To compute the value of $cos(\theta)$, we look up the display surface normal in the normal map and take the dot product of the normal and the unit vector from X to the projector center. Composing the rest of the model terms together, we do a final look-up in the camera response texture to get the final predicted intensity for the pixel. The shader program then returns the predicted intensity in one color channel of the predicted image with the other two channels left black. This allows us to read back the predicted image to the CPU as a single channel texture, greatly improving read-back efficiency.

Figure 3 shows a captured camera image and a predicted image rendered using our technique. We computed the difference between these images to be 15.1 intensity levels on average per pixel with a standard deviation of 3.3. The images appear very similar, with the most significant difference being the higher contrast in the rendered image. The rendered image is also slightly darker. We attribute both of these differences to indirect scattering effects caused by the complex display surface geometry being present in the camera image, but not reproduced in the predicted image.

5. Continuous Projector Calibration

To track the projector motion in real-time using the predicted image, we first detect a set of features in the captured camera image during each frame. As described previously, by matching features between the captured image and the predicted image, we indirectly obtain a set of 2D-3D point projector point correspondences that allows us to estimate the pose of the projector. For feature detection in the captured image, we use Intel's OpenCV library implementation of the feature detection algorithm introduced by

Figure 3. An actual image captured by a camera (left) and a rendered prediction of the camera image (right).

Shi and Tomasi [19]. To obtain correspondences for the detected features in the predicted image, we use pyramidal Lucas-Kanade tracking [12, 4], also part of OpenCV.

To transform these correspondences into 2D-3D projector point correspondences, we use the same geometry map used in rendering the predicted image, which stores the mapping between camera pixels and 3D points on the display surface. To obtain the 3D correspondences, we perform a look-up in the geometry map for each feature in the captured image. To obtain the 2D correspondences for these points, we reverse the mapping from projected image pixels to predicted image pixels. This is done by consulting the geometry map using the feature locations in the predicted image and projecting the resulting points into the projected image using the projector calibration.

5.1. Pose Estimation

We assume that any motion of the projector does not affect its intrinsic calibration, requiring only the 6 parameters comprising the extrinsic calibration (position and orientation) to be re-estimated. To calculate the projector pose from the 2D-3D correspondences, we use a common technique from analytic photogrammetry described in Haralick [11].

The technique takes as input a set of 2D-3D correspondences as well as an initial estimate of the pose and minimizes the sum of squared reprojection errors using a nonlinear least-squares technique. While the need for an initial guess is a limitation of the technique, it has the advantage of being able to compute the correct pose in situations where a linear technique will fail, such as when all 3D correspondences are coplanar. We have found that using the previous pose of the projector as an initial guess for the current pose is adequate for convergence even when the projector pose is changing rapidly.

5.2. Outliers

We have found it essential to incorporate RANSAC into the pose estimation to prevent false correspondences from affecting the pose estimation results. In our RANSAC approach, we estimate the projector pose using three correspondences selected at random and record the number of correspondences whose resulting reprojection error is less than 10 projector pixels as inliers. We perform this iteration repeatedly until there is a 99% chance that at least one interation has chosen a set of three correct corrspondences. We then take the largest inlier set over all iterations and perform a final pose estimate.

5.3. Filtering

Since we do not synchronize the projector and camera, it is possible for the captured and predicted images to be out of step by a frame. This introduces error into the correspondences used to calibrate the projector and can lead to instability if the pose estimates are not filtered.

In our experimentation, we have found that by limiting the amount of change that is made to the pose estimate each frame, any instability caused by an unsynchronized camera and projector can be removed. After the change in pose has been calculated, we add it to the current pose estimate at 10% of its original scale. We have found this to ensure tracking stability while still providing a responsive system.

6. Results

We tested the tracking ability of our system using two dynamic applications. The first application displays a rotating panorama of a real environment and contains many strong features. The second application is a virtual flight simulator that displays rendered 3D geometry and is relatively sparse in strong features.

Figure 4. **a)** An image captured by a camera. **b)** An image rendered to predict the captured image. **c)** The projected image that was captured. **d)** A contrast-enhanced difference image of the captured and predicted images.

Imagery	Avg. Feature Count	Avg. % Inliers
Panorama(front)	191	81
Panorama(back)	197	84
Flight Simulator	36	60

Table 1. Feature matching performance using **both geometric and radiometric prediction** of the captured image.

Imagery	Avg. Feature Count	Avg. % Inliers
Panorama(front)	183	38
Panorama(back)	192	46
Flight Simulator	13	30

Table 2. Feature matching performance using **only geometric prediction** of the captured image.

To test the feature detection and matching capability of the system, we recorded the number of features successfully matched between the predicted and camera images and the number of these matches found to be correct by RANSAC in each frame. Table 1 shows the results we collected over 1000 frames for both applications. The maximum number of features to detect and match was limited to 200 for this experiment, which we have found to be more than adequate for tracking.

While the results are considerably better for the panorama application, we were not able to notice a visible difference in the quality of the tracking results. This is due to the lower number of matched features in the flight simulator being sufficient to accurately calculate the projector pose and the success of RANSAC in removing the incorrect correspondences.

To estimate the improvement in feature matching gained by performing radiometric prediction, we ran the same tests using predicted images generated using only geometric pre-

diction. The predicted images were converted to grayscale using the NTSC coefficients for color to grayscale conversion. The results of this experiment are present in Table 2. Note that during this experiment, tracking for the flight simulator was lost after only 6 frames.

The performance of the tracker is heavily dependent on the number of features that the system attempts to detect and match each frame. Setting the maximum number of features to detect and match at 75, we were able to obtain excellent tracking results and measured the performance of the tracker to be approximately 27 Hz for both applications.

7. Summary and Future Work

We have described a new technique that enables a projector to be tracked in real-time without affecting the quality of the projected imagery. This technique allows dynamic repositioning of the projector without display interruption. By matching features between the projector and a static camera, the pose of the projector is estimated each frame. Since obtaining correspondences between the projector and camera can be difficult, we obtain these correspondences indirectly by matching between the camera image and an image rendered using the current calibration information to predict the image the camera will capture. We describe a simple radiometric model that can easily be implemented on graphics hardware to produce this predicted image.

Our current technique has certain limitations we would like to overcome in future work. Our radiometric calibration process currently requires a uniform display surface albedo and that there be a portion of the display surface to project on that does not introduce substantial indirect scattering effects. A futher limitation is the reliance on the presence of features in the user imagery that are suitable for matching. We believe, however, that exposing the camera for a short frame interval will expose additional features in the user imagery when using a DLP projector since small differences in intensity can be magnified greatly by the mirror flips that occur.

8. Acknowledgements

The authors wish to thank Herman Towles and Rick Skarbez for their insights and suggestions on this paper.

References

[1] M. Ashdown, M. Flagg, R. Sukthankar, and J. Rehg. A flexible projector-camera system for multi-planar displays, 2004.

[2] O. Bimber, A. Emmerling, and T. Klemmer. Embedded entertainment with smart projectors. *IEEE Computer*, 38(1):56–63, 2005.

[3] O. Bimber, G. Wetzstein, A. Emmerling, and C. Nitschke. Enabling view-dependent stereoscopic projection in real environments. In *4th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 14–23, 2005.

[4] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. In *Technical Report, Intel Corporation*, 1999.

[5] D. Cotting, M. Naef, M. Gross, and H. Fuchs. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *International Symposium on Mixed and Augmented Reality*, pages 100–109, 2004.

[6] D. Cotting, R. Ziegler, M. Gross, and H. Fuchs. Adaptive instant displays: Continuously calibrated projections using per-pixel light control. In *Eurographics*, pages 705–714, 2005.

[7] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *Computer Graphics*, 31(Annual Conference Series):369–378, 1997.

[8] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial grayscale. *Journal of the Society for Information Display*, 17(2):75–77, 1976.

[9] D. A. Forsyth and J. Ponce. *Computer Vision A Modern Approach*. Prentice Hall, 1st edition, 2003.

[10] M. D. Grossberg, H. Peri, S. K. Nayar, and P. N. Belhumeur. Making one object look like another: Controlling appearance using a projector-camera system. In *IEEE Computer Vision and Pattern Recognition*, volume 1, pages 452–459, 2004.

[11] R. Haralick and L. Shapiro. *Computer and Robot Vision*, volume 2. 1993.

[12] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[13] S. K. Nayar, H. Peri, M. D. Grossberg, and P. N. Belhumeur. A projection system with radiometric compensation for screen imperfections. In *ICCV Workshop on Projector-Camera Systems (PROCAMS)*, October 2003.

[14] P. Quirk, T. Johnson, R. Skarbez, H. Towles, F. Gyarfas, and H. Fuchs. Ransac-assisted display model reconstruction for projective display. In *Emerging Display Technologies*, 2006.

[15] A. Raij and M. Pollefeys. Auto-calibration of multi-projector display walls. In *International Conference on Pattern Recogniton*, 2004.

[16] R. Raskar, M. S. Brown, R. Yang, W.-C. Chen, G. Welch, H. Towles, W. B. Seales, and H. Fuchs. Multi-projector displays using camera-based registration. In *IEEE Visualization*, pages 161–168, 1999.

[17] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ilamps: Geometrically aware and selfconfiguring projectors. In *ACM SIGGRAPH*, 2003.

[18] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Eurographics Workshop on Rendering*, 2001.

[19] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[20] R. Yang and G. Welch. Automatic and continuous projector display surface estimation using every-day imagery. In *9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2001.

IPT-EGVE Symposium (2007)
B. Fröhlich, R. Blach, and R. van Liere (Editors)

A Unified Multi-Surface, Multi-Resolution Workspace with Camera-Based Scanning and Projector-Based Illumination

T. Johnson and H. Fuchs

University of North Carolina - Chapel Hill, USA

Abstract

Research in the area of projector- and camera-augmented office environments has demonstrated the use of cameras as desktop scanning devices, shown the benefits of using multiple display devices for focus and context information, and advocated display on multiple surfaces for visualization of multi-dimensional data. In this paper, we describe how the implementation of these ideas can be unified to simplify the creation of a workspace that combines them.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Display algorithms

1. Introduction

Research in projector- and camera-augmented office environments has made it possible to create large digital desktops that allow digital documents to be interacted with as if they were real documents. The digital desk [Wel93] blurred the line between real and digital content by allowing digital documents displayed on a desk surface to be marked with a pen as if they were real. It also demonstrated how cameras in the office could be used for desktop scanning, allowing user-selected portions of real documents to be copied and pasted into digital documents directly on the desk.

Other researchers have experimented with combining multiple display devices to create focus plus context displays [BGS01, AR03]. In this type of system, one display provides high spatial resolution for tasks such as text editing, while another display provides context information at a lower spatial resolution, e.g. other documents that may be of interest, but not currently being edited. This context information is useful in many applications.

By extending the digital desktop beyond just the desktop itself and onto other surfaces as well, the workspace can also be used to visualize 3D content. Ashdown et al. [AFSR04] describe how this can be accomplished using a multi-planar surface. In this work, an automatic technique for calculating mutually consistent rectifying homographies between each plane and the projector is described.

We imagine an office environment that combines all of these ideas in a single workspace. A conceptual illustration of our vision for this new type of workspace is shown in Figure 1. In this paper, we describe techniques for calibration

Figure 1: *Multi-surface workspace concept. (Andrei State)*

and rendering that allow these previous ideas from projector- and camera-augmented office environments to be combined to create a multi-surface, multi-resolution workspace with camera-based scanning and projector-based illumination. We show that techniques from multi-projector displays can provide a robust calibration procedure and simplify the task of combining these ideas in a common framework.

2. The Workspace

The workspace we have created is reminiscent of an office cubicle. Our environment, as seen in Figure 3, consists of a flat desktop abutted on two sides by vertical walls. An LCD panel is also embedded within the surface of the desktop. A projector-camera module consisting of a projector and two greyscale cameras is positioned a short distance away to illuminate the surface of the desk as well as the two walls.

3. Calibration

Before imagery is displayed on the workspace, a simple calibration procedure is performed. This determines the information needed to project imagery onto the surfaces of the workspace in a way that compensates for the orientation of the surfaces with respect to the projector. Instead of a homography-based approach, we perform a full 3D calibration, which easily allows content to be displayed across multiple surfaces. The procedure estimates a polygonal model of the workspace geometry as well as a projection matrix for the projector, both cameras, and the LCD panel. This determines an invertible mapping from 2D pixel locations in each device to 3D locations on the surfaces of the workspace.

Our calibration process begins by projecting encoded structured light patterns that are observed by the cameras. Decoding of these structured light patterns allows precise image correspondences between the cameras to be obtained. Since the projector and cameras are rigidly attached, we assume the devices have been pre-calibrated using an existing technique such as [Zha00, RWC*98]. The camera calibration can then be used to triangulate each correspondence into a 3D point, forming a 3D point-cloud representation of the surfaces of the workspace.

To construct a polygonal model from this point-cloud representation, we use the RANSAC-based plane-fitting algorithm described in Quirk [QJS*06], which is robust against noise and outlying points resulting from false stereo matching. The algorithm is used to find the three most dominant planes in the point-cloud data, which are then intersected to form a simple polygonal mesh. This plane intersection process allows the wall corners of the workspace, where calibration errors will be most apparent, to be accurately estimated.

We treat the LCD screen as a projector and display the same structured light patterns. This results in a point-cloud representation of the plane of the LCD screen in the same coordinate system as the projector-camera module and the polygonal model of the workspace. We then calculate a projection matrix relating the 3D points on the surface of the LCD screen to their 2D pixel coordinates in the LCD image. While this mapping could also be accomplished using a homography, a 3D calibration allows the LCD screen to be handled in the same way as the projector during rendering.

The standard technique for calculating projection matrices from 2D-3D correspondences is the direct linear transform (DLT) [AAK71]. In the case of the LCD screen, the 3D points are coplanar, and the DLT algorithm will fail. Knowledge of the center-of-projection (COP) is however sufficient to fully determine a projection matrix. The choice of COP is arbitrary in this case, so we choose it to lie some distance along the normal of the LCD plane.

4. Rendering

In this section, we describe how it is possible to provide the user with content displayed on only a single surface with content displayed on multiple surfaces in a unified way using a single technique - two-pass rendering [RWC*98]. We also describe how this same technique can be used to correct image distortion when using a camera for desktop scanning.

4.1. Two-Pass Rendering

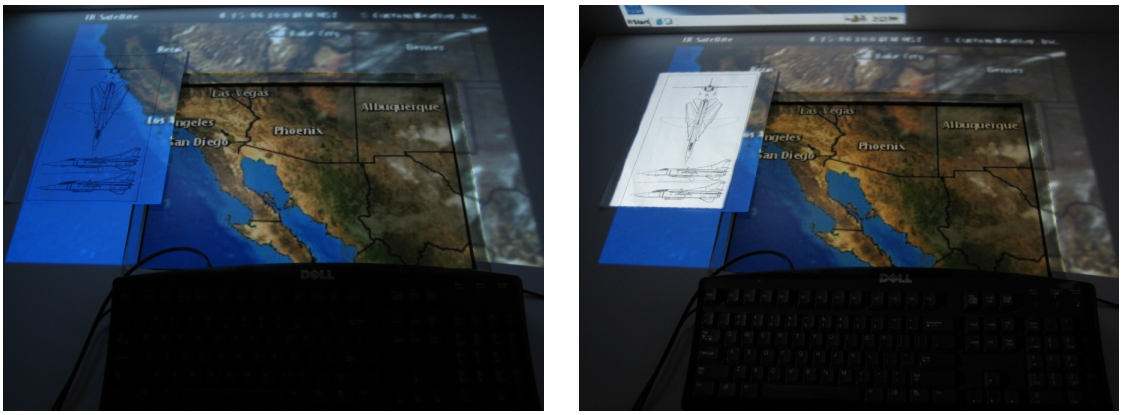
The basic approach behind two-pass rendering is to take a desired image to be observed on a projection surface and determine the necessary warping of this desired image to compensate for the geometry of the projection surface.

The desired image is rendered in pass one. This can be an image generated from a 3D graphics application or simply an image of a 2D windowed application such as a word processor. The warping of this image occurs in the second pass using projective texturing, where the image is projected onto the geometry of the projection surface from the user's viewpoint. The textured surface geometry is then rendered from the perspective of the projector using its projection matrix.

4.2. Single-Surface Content

Some types of imagery, such as that of a word processor, are best displayed to the user on a single surface of the workspace. While this content can be made to look geometrically correct across multiple surfaces from the user's viewpoint, the notion that this type of content is flat can conflict with the stereo cues the user receives and be disturbing.

Using two-pass rendering, we can easily determine how to display a window in order to align it with the natural coordinate system of one of the workspace surfaces. When a window is opened on one of the surfaces, the four coplanar corners of the window are used to calculate a projection matrix with a COP located a short distance from the center of



a) b)

Figure 2: ***a)*** *Undesired projection on a real document.* ***b)*** *Neutral illumination on a document and keyboard.*

the window along the plane normal. This is identical to the calibration process we described for the LCD screen.

During rendering, this matrix is used as the projective texturing matrix in pass-two to project the window contents onto the geometry of the workspace model before it is rendered from the perspective of the projector. Note that this process for displaying content on a single surface is independent of the location of the viewer.

4.3. Multi-Surface Content

The two-pass rendering algorithm also supports the display of imagery across multiple surfaces. The workspace can then also be used as a convenient medium for visualizing content such as 3D models or virtual data sets. The rendering process used to display this content is identical to that used to display single-surface content except that the COP of the projective texturing matrix is constrained to be the viewpoint of the user, which can be maintained by a separate tracking system.

4.4. Desktop Scanning

Using the projector-camera module, it is also possible to create a desktop scanner, allowing the user to copy from real documents and paste into digital ones without the need to leave his desk. This technology was first demonstrated as part of the digital desk [Wel93]. Here, we describe how this technology can be implemented in the context of a multi-surface workspace to allow documents located on any surface of the workspace to be copied.

Using a selection rectangle, which is handled as a single-surface window that follows the motion of the mouse, the user selects the portion of a document he wishes to copy. The projector then illuminates the selected area to improve its brightness while the camera captures an image. Two steps now remain - the user-selected portion of the camera image must be segmented from the rest of the image, and the distortion caused by the orientation of the camera with respect to the surface must be removed.

Both of these steps can be accomplished simultaneously using two-pass rendering. We set the desired image to be the captured camera image and use the camera's projection matrix as the projective texturing matrix. The geometry that is projectively textured and rendered is a quad formed from the corners of the selection rectangle. This quad is then rendered with the projection matrix of the selection rectangle window.

4.5. Projector-Based Illumination

There are often objects present in the office environment on which it may not be desirable to have imagery projected. As shown in Figure 2a, when documents have imagery projected on them, it can be difficult to distinguish the document from the projected imagery. This is especially troubling for documents containing text, which are made difficult to read.

We have created a simple interface that allows the user to select the corners of a quad where uniform white light should be displayed by the projector. In Figure 2b, the imagery projected on the document has been replaced with uniform white light from the projector, allowing the contents of the document to be clearly seen. The keyboard has also been illuminated by the projector in this image.

5. Results

We have implemented an application allowing windows containing various types of content to be opened on any surface of the workspace. The windows are simply static frames used to demonstrate the calibration and rendering and are not

tied to any application. Figure 3 is an image of the application running. Using our rendering process, when a window overlaps the LCD screen, the overlapping window contents are automatically displayed in high resolution on the LCD screen and registered to the surrounding projected imagery. The satellite in the image demonstrates the ability to simultaneously view content on multiple surfaces.

Figure 4 shows a hard-copy document and the resulting digitized version selected by the user to be scanned. The distortion involved in capturing an image of the document from the perspective of the camera has been eliminated.

Figure 3: *A multi-surface, multi-display workspace combining both single- and multi-surface content.*

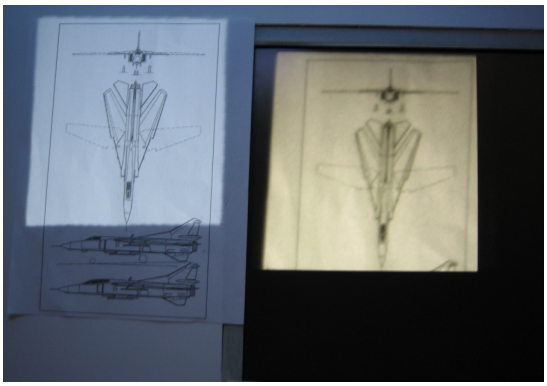


Figure 4: *Desktop scanning using a projector-camera module.*

6. Conclusions and Future Work

We have described a framework that allows previously separate ideas from projector- and camera-augmented office environments to be combined into a single workspace. Our system includes a multi-planar surface, a projector-camera module, and an LCD screen and allows single- and multi-surface content to be displayed in a unified way. Using this framework, the LCD screen is calibrated to the projector and its imagery automatically registered to the projected imagery to provide an area for tasks requiring high resolution. We also describe how the cameras of the projector-camera module can be used for desktop scanning within the same framework to further simplify implementation.

We have focused solely on rendering and calibration issues in this paper, but the potential for new interaction techniques should also be investigated. We think this type of display has great potential for remote collaboration with the wall surfaces being used as a window to a remote environment. In the future, we hope to replace the need for manual selection of objects that should only be illuminated by white light with an automatic technique that detects objects occluding projected imagery.

7. Acknowledgments

The authors would like to thank Herman Towles and John Thomas for building the workspace used in this project, as well as Andrei State for his conceptual artwork. This work was partially supported by the VIRTE program under Office of Naval Research award number N00014-03-1-0589.

References

[AAK71] ABDEL-AZIZ Y., KARARA H.: Direct linear transformation into object space coordinates in close-range photogrammetry. In *Symposium on Close-Range Photogrammetry* (1971), pp. 1–18.

[AFSR04] ASHDOWN M., FLAGG M., SUKTHANKAR R., REHG J.: A flexible projector-camera system for multi-planar displays. In *CVPR* (2004).

[AR03] ASHDOWN M., ROBINSON P.: The escritoire: A personal projected display. In *11th International Conference in Central Europe on Computer Graphics* (2003), pp. 33–34.

[BGS01] BAUDISCH P., GOOD N., STEWART P.: Focus plus context screens: combining display technology with visualization techniques. In *UIST* (2001), pp. 31–40.

[QJS*06] QUIRK P., JOHNSON T., SKARBEZ R., TOWLES H., GYARFAS F., FUCHS H.: Ransac-assisted display model reconstruction for projective display. In *Emerging Display Technologies* (2006).

[RWC*98] RASKAR R., WELCH G., CUTTS M., LAKE A., STESIN L., FUCHS H.: The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH* (New York, NY, 1998), ACM Press, pp. 179–188.

[Wel93] WELLNER P.: Interacting with paper on the DigitalDesk. *Communications of the ACM 36*, 7 (1993).

[Zha00] ZHANG Z.: A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 11 (2000), 1330–1334.

BIBLIOGRAPHY

[Abdel-Aziz and Karara, 1971] Abdel-Aziz, Y. and Karara, H. (1971). Direct linear transformation into object space coordinates in close-range photogrammetry. In *Symposium on Close-Range Photogrammetry*, pages 1–18.

[Bandyopadhyay et al., 2001] Bandyopadhyay, D., Raskar, R., and Fuchs, H. (2001). Dynamic shader lamps: Painting on movable objects. In *IEEE and ACM International Symposium on Augmented Reality*.

[Bar-Shalom and Li, 1998] Bar-Shalom, Y. and Li, X. R. (1998). *Estimation and Tracking: Principles, Techniques, and Software*. YBS Publishing.

[Baudisch et al., 2001] Baudisch, P., Good, N., and Stewart, P. (2001). Focus plus context screens: combining display technology with visualization techniques. In *UIST*, pages 31–40.

[Bhasker et al., 2006] Bhasker, E. S., Sinha, P., and Majumder, A. (2006). Asynchronous distributed calibration for scalable and reconfigurable multi-projector displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1101–1108.

[Bimber et al., 2005] Bimber, O., Wetzstein, G., Emmerling, A., and Nitschke, C. (2005). Enabling view-dependent stereoscopic projection in real environments. In *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, pages 14–23.

[Blair and Peyton, 1993] Blair, J. and Peyton, B. (1993). *An Introduction to Chordal Graphs and Clique Trees*. Graph Theory and Sparse Matrix Computations. Springer-Verlag.

[Bouguet, 1999] Bouguet, J.-Y. (1999). Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation.

[Bouguet, 2008] Bouguet, J.-Y. (2008). Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[Brown, 1971] Brown, D. (1971). Close-range camera calibration. *Photometric Engineering*, 37(8):855–866.

[Buehler et al., 2001] Buehler, C., Bosse, M., McMillan, L., Gortler, S., , and Cohen, M. (2001). Unstructured lumigraph rendering. In *Proceedings of ACM SIGGRAPH*.

[Chen et al., 2002] Chen, H., Sukthankar, R., Wallace, G., and Li, K. (2002). Scalable alignment of large-format multi-projector displays using camera homography trees. In *Proceedings of IEEE Visualization (Vis2002)*.

[Clipp et al., 2007] Clipp, B., Welch, G., Frahm, J.-M., and Pollefeys, M. (2007). Structure from motion via a two-stage pipeline of extended kalman filters. In *British Machine Vision Conference*.

[Cotting et al., 2004] Cotting, D., Naef, M., Gross, M., and Fuchs, H. (2004). Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 100–109.

[Cotting et al., 2005] Cotting, D., Ziegler, R., Gross, M., and Fuchs, H. (2005). Adaptive instant displays: Continuously calibrated projections using per-pixel light control. In *Eurographics*, pages 705–714.

[Cowell et al., 1999] Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag.

[Dellaert et al., 2005] Dellaert, F., Kipp, A., and Krauthausen, P. (2005). A multifrontal qr factorization approach to distributed inference applied to multi-robot localization and mapping. In *American Association for Artificial Intelligence (AAAI)*, pages 1261–1266.

[Faugeras and Toscani, 1987] Faugeras, O. and Toscani, G. (1987). Camera calibration for 3d computer vision. In *International Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, pages 240–247.

[Fischler and Bowles, 1981] Fischler, M. and Bowles, R. (1981). RANdom SAmple Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 6(24):381–395.

[Grewal and Andrews, 2008] Grewal, M. S. and Andrews, A. P. (2008). *Kalman Filtering Theory and Practice Using MATLAB*. John Wiley & Sons, Inc.

[Grundhöfer et al., 2007] Grundhöfer, A., Seeger, M., Häntsch, F., and Bimber, O. (2007). Coded projection and illumination for television studios. In *Bauhaus-University Weimar, Technical Report 843*.

[Gupta and Jaynes, 2006] Gupta, S. and Jaynes, C. (2006). The universal media book: tracking and augmenting moving surfaces with projected information. In *Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06)*.

[Haralick and Shapiro, 1993] Haralick, R. and Shapiro, L. (1993). *Computer and Robot Vision*, volume 2. Prentice Hall.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition.

[Heikkil and Silvn, 1997] Heikkil, J. and Silvn, O. (1997). A four-step camera calibration procedure with implicit image correction. In *CVPR*.

[Johnson and Fuchs, 2007a] Johnson, T. and Fuchs, H. (2007a). Real-time projector tracking on complex geometry using ordinary imagery. In *IEEE International Workshop on Projector-Camera Systems (PROCAMS)*.

[Johnson and Fuchs, 2007b] Johnson, T. and Fuchs, H. (2007b). A unified multi-surface, multi-resolution workspace with camera-based scanning and projector-based illumination. In *Eurographics Symposium on Virtual Environments/Immersive Projection Technology Workshop*.

[Kalman, 1960] Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, pages 35–45.

[Kannala and Brandt, 2006] Kannala, J. and Brandt, S. (2006). A generic camera model and calibration method for conventional, wide-angle and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):451–460.

[Konieczny et al., 2005] Konieczny, J., Shimizu, C., Meyer, G., and Colucci, D. (2005). A handheld flexible display system. In *IEEE Visualization*.

[Longuet-Higgins, 1981] Longuet-Higgins, H. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2).

[Lucas and Kanade, 1981] Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130.

[Majumder, 2002] Majumder, A. (2002). Properties of color variation across a multi-projector display. In *SID Eurodisplay*.

[Majumder et al., 2000] Majumder, A., He, Z., Towles, H., and Welch, G. (2000). Color calibration of projectors for large tiled displays. In *IEEE Visualization*.

[Majumder and Stevens, 2002] Majumder, A. and Stevens, R. (2002). LAM: Luminance attenuation map for photometric uniformity in projection based displays. In *ACM Virtual Reality and Software Technology (VRST)*.

[Maybeck, 1979] Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, Inc.

[Myers and Tapley, 1976] Myers, K. and Tapley, B. (1976). Adaptive sequential estimation with unknown noise statistics. *IEEE Transactions on Automatic Control*, 21:520–523.

[Posdamer and Altschuler, 1982] Posdamer, J. L. and Altschuler, M. D. (1982). Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1–17.

[Pothen and Sun, 1992] Pothen, A. and Sun, C. (1992). Distributed multifrontal factorization using clique trees. In *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pages 34–40.

[Quirk et al., 2006] Quirk, P., Johnson, T., Skarbez, R., Towles, H., Gyarfas, F., and Fuchs, H. (2006). Ransac-assisted display model reconstruction for projective display. In *Emerging Display Technologies*.

[Raij et al., 2003] Raij, A., Gill, G., Majumder, A., Towles, H., and Fuchs, H. (2003). Pixelflex2: A comprehensive, automatic, casually-aligned multi-projector display. In *IEEE International Workshop on Projector-Camera Systems (PROCAMS-2003)*.

[Raij and Pollefeys, 2004] Raij, A. and Pollefeys, M. (2004). Auto-calibration of multi-projector display walls. In *International Conference on Pattern Recognition*.

[Rao and Durrant-Whyte, 1991] Rao, B. and Durrant-Whyte, H. (1991). Fully decentralised algorithm for multisensor kalman filtering. In *Control Theory and Applications, IEE Proceedings D*.

[Raskar et al., 1999a] Raskar, R., Brown, M., Yang, R., Chen, W.-C., Welch, G., Towles, H., Seales, B., and Fuchs, H. (1999a). Multi-projector displays using camera-based registration. In *Proceedings of Visualization '99*, pages 161–168, 522, San Francisco, CA, USA. IEEE.

[Raskar et al., 1999b] Raskar, R., Brown, M. S., Yang, R., Chen, W.-C., Welch, G., Towles, H., Seales, W. B., and Fuchs, H. (1999b). Multi-projector displays using camera-based registration. In *IEEE Visualization*, pages 161–168.

[Raskar et al., 2003] Raskar, R., van Baar, J., Beardsley, P., Willwacher, T., Rao, S., and Forlines, C. (2003). iLamps: Geometrically aware and selfconfiguring projectors. In *Proceedings ACM SIGGRAPH*.

[Raskar et al., 1998] Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. (1998). The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 179–188, New York, NY, USA. ACM Press.

[Raskar et al., 2001] Raskar, R., Welch, G., Low, K.-L., and Bandyopadhyay, D. (2001). Shader lamps: Animating real objects with image-based illumination. In *Eurographics Workshop on Rendering*.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle.

[Smith et al., 1989] Smith, R., Self, M., and Cheeseman, P. (1989). Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*.

[Surati, 1999] Surati, R. (1999). Scalable self-calibrating display technology for seamless large-scale displays. *PhD thesis, Massachusetts Institute of Technology*.

[Underkoffler and Ishii, 1998] Underkoffler, J. and Ishii, H. (1998). Illuminating light: An optical design tool with a luminous-tangible interface. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*.

[Walter and Leonard, 2004] Walter, M. and Leonard, J. (2004). An experimental investigation of cooperative SLAM. In *Proceedings of the Fifth IFAC Symposium on Intelligent Autonomous Vehicles*.

[Welch and Bishop, 1997] Welch, G. and Bishop, G. (1997). SCAAT: Incremental tracking with incomplete information. *Computer Graphics*, 31(Annual Conference Series):333–344.

[Welch and Bishop, 2006] Welch, G. and Bishop, G. (2006). An introduction to the kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill.

[Yang et al., 2001] Yang, R., Gotz, D., Hensley, J., Towles, H., and Brown, M. (2001). Pixelflex: A reconfigurable multi-projector display system. In *IEEE Visualization*.

[Yang and Welch, 2001] Yang, R. and Welch, G. (2001). Automatic and continuous projector display surface estimation using every-day imagery. In *9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*.

[Zhou et al., 2008] Zhou, J., Wang, L., Akbarzadeh, A., and Yang, R. (2008). Multi-projector display with continuous self-calibration. In *Workshop on Projector-Camera Systems (PROCAMS)*.

[Zollmann et al., 2006] Zollmann, S., Langlotz, T., and Bimber, O. (2006). Passive-active geometric calibration for view-dependent projections onto arbitrary surfaces. In *Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR*.