Shadow Removal from Multi-Projector Displays via Three-Dimensional Modeling and Object Tracking

Samuel Audet

Master of Engineering

Department of Electrical and Computer Engineering

McGill University

Montréal, Québec

November 2007

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Engineering

ABSTRACT

When a person stands between a projector and a display surface a shadow occurs. We show a method to perform shadow removal using camera-based object tracking and a three-dimensional model of a room, which includes cameras, projectors, and flat surfaces. For tracking people, although we use cameras, other methods would work. To obtain the model, we adapted and use existing geometric calibration methods. With the tracking and model information, our algorithm finds in the projector image the region that is occluded by a person. Another projector can then automatically fill the region with equal intensity and identical color to that of the occluded projector. We found that calibration and tracking were accurate, and that the system could correctly and efficiently perform shadow removal, providing a more appealing visual experience to users of multi-projector displays.

SOMMAIRE

Lorsqu'une personne se tient entre un projecteur et une surface d'affichage une ombre apparaît. Nous démontrons une méthode pour éliminer les ombres en utilisant la poursuite d'objets par caméras et un modèle tridimensionnel d'une pièce, qui inclut les caméras, les projecteurs et les surfaces plates. Pour la poursuite de personnes, quoique nous utilisons des caméras, d'autres méthodes fonctionneraient. Pour obtenir le modèle, nous avons adapté et utilisons des méthodes de calibration géométrique existantes. Avec l'information sur la poursuite et le modèle, notre algorithme trouve dans l'image d'un projecteur la région qui est occluse par une personne. Un autre projecteur peut alors automatiquement remplir la région avec les mêmes intensités et couleurs que celles du projecteur occlus. Nous avons trouvé que la calibration et la poursuite étaient exactes et que le système était capable d'éliminer correctement et efficacement les ombres, procurant une expérience visuelle plus attrayante aux utilisateurs de l'affichage à projecteurs multiples.

ACKNOWLEDGEMENTS

First and foremost, I am deeply grateful for the research opportunities, the numerous insights and valuable wisdoms that I received from my advisor Jeremy R. Cooperstock as part of the Shared Reality Laboratory. I thank him for his patience and understanding during my stay as well as for the editorial help with my thesis. Next, this work would not have been possible without the generous scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC). Also many thanks to Linda Cooper for her wonderful seminars on writing science articles, which gave me the necessary knowledge to write better and more clearly. I enjoyed as well the many useful discussions I had with Stéphane Pelletier. Finally, thank you to all the professors I had the pleasure to work with, in addition to my advisor, for the precious knowledge I gained from their lectures and courses: Tal Arbel, Martin D. Levine, Godfried T. Toussaint, Paul Tupper, and Yon Visell.

The great picture from Mars is courtesy of NASA/JPL-Caltech.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
Introduction

We propose a new method to perform shadow removal from multi-projector displays, but before going into the technical details, we first provide an overview of the field to which this research applies, the outstanding issues and a review of other related works currently available in the literature.

1.1 Field of Research

With respect to the field of research, our work falls mainly into the domain of *virtual reality*, a technology that allows people to interact with a 3D environment generated and displayed by computers. Although the relevance of many systems rise and fall, the interest in the *CAVE* (Cave Automatic Virtual Environment) remains strong as can be seen from the large number of installations in universities and other institutions worldwide [10, 47]. Cruz-Neira *et al.* developed the first CAVE in 1992 [11, 12] and it has since become a major player in the field of immersive virtual reality, where computer-generated graphics completely encompass a user's field of view. A CAVE consists of a room equipped with three or more screens enclosing a space in the middle of the room, where projectors display images from behind the screens traditionally with the help of mirrors, a technique known as *back projection*. Unlike other methods, such as virtual reality goggles (head-mounted displays), when using a CAVE, multiple users can to a certain extent naturally interact with each other while experiencing the same virtual environment at the same time. Users can also easily manipulate real objects within the confines of the CAVE. For these reasons, this kind of virtual reality technology has gained great interest in the scientific and engineering communities allowing teams of researchers to analyze data more quickly and easily. Interest in such technology is also growing for its many potential applications to telepresence or *shared reality* [8], and multimedia entertainment. For example, related and recently developed immersive technologies such as Cisco TelePresence receive millions of dollars in research and development [40].

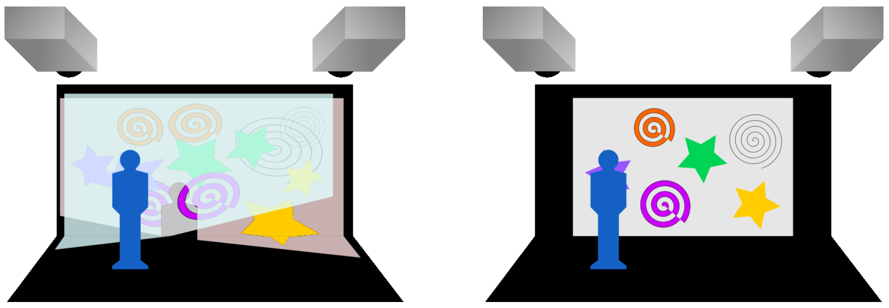
1.2 Outstanding Issues

Unfortunately, it is not practical to mount and unmount a CAVE whenever needed, and consequently room space usually needs to be reserved. An alternative to back projection used by the CAVE is *front projection*, such as used in the Office of the Future [31]. In a front projection configuration, projectors are mounted in front of the screen and they do not need mirrors or screens in the middle of a room. Front projection installations thus use less space than equivalent rear projection configurations, and by its nature one projector can cover more surface than a monitor. Considering how commonly people use projectors in this fashion, front projection can considerably increase the attractiveness of CAVE-like environments.

However, front projection has disadvantages as well. First, projectors are installed according to the configuration of the room and are not necessarily placed right in front of the display surface, causing geometric distortion. Second, a casually chosen display surface (curtains, painted wall, etc.) might not be flat and might not perfectly reflect projector light. Third, when projecting an image at an angle from the display surface, parts of the image might be out of focus. Fortunately, researchers have done extensive work in developing methods to compensate for these disadvantages. The Metaverse by Jaynes *et al.* [21] and the system developed by Bimber *et al.* [3] are good examples of the latest developments in this direction. Unfortunately, even with these sophisticated methods, when a person moves in between a projector and the display surface a shadow still occurs. This is one important remaining problem of front projection displays, negatively affecting the experience of a user [35, 36].

In many situations, we can manage shadows by appropriately positioning projectors and constraining the allowed locations at which people can stand. When this is not possible, such as in small rooms, we can use techniques where multiple projectors are configured so they produce the same image at the same place on the display surface. This is also known as *Passive Virtual Rear Projection*. The image thus remains visible even if a person occludes one projector. This configuration can be achieved by transforming the images before their projection so they match with the reference frame [30, 31, 43, 45]. For a flat display surface, the transformation may be a simple homography, as we explain later in this work. Color correction and multi-focal projection are also desirable, although outside the scope of our work.

With the previous approach, although the image remains visible during occlusion, the resulting differences in display intensity remain perceptible. To compensate, shadow removal methods determine the display region in which a



(a) Before corrections.
(b) After corrections.

Figure 1–1: Ideal front projection where distortion, colors and shadows are all corrected.

shadow occurs. Another projector can then fill the region with equal intensity and identical color to that of the occluded projector, as in the ideal case depicted in Figure 1–1. This is also known as *Active Virtual Rear Projection*. However, as described in the next section, current methods suffer from a number of limitations.

1.3 Literature Review

All shadow removal methods for front projection have one thing in common. They all use at least two projectors to cover any given region of the display surface. Still, current methods can be divided into two main families: shadow detection and occlusion detection. In the former, we use cameras to watch the display surface and detect regions with undesirable lowering of intensity caused by shadows. We compensate for these changes by increasing the output of the projectors to recover the desired brightness in those regions. On the other hand, with occlusion detection, the idea is to use other information instead of the display content and to segment out the occlusion occurring in front of a projector. These methods need to be insensitive to a changing display and attempt the operation by using color transfer functions to predict the background, by modelling the background in a generic manner, by using synchronized camera-projector systems to produce patterns imperceptible to the human eye, or by using infrared cameras, which are insensitive to projector light. When the occlusion in front of a projector is detected in this fashion, we can directly use the information to create masks for the projectors.

1.3.1 Shadow Detection Methods

The first shadow removal methods that were developed detect shadows on the display surface [6, 18, 22, 23, 34]. These methods work by comparing the image displayed by the projectors to the image as seen by a set of cameras, after compensating for geometric and radiometric color differences between them. Therefore, this operation can only remove shadows after they appear and becomes increasingly complex in the case of video projection, since to compare correctly we need to know the precise delay incurred in the projectors, in the cameras, and in processing. An additional constraint is that we need to be able to reconstruct an unoccluded view of the entire display surface using the cameras, or the system cannot function properly. Furthermore, flickering may result when the system no longer observes a shadow and incorrectly assumes the region to be unoccluded.

Sukthankar *et al.* [34] are the first to have tackled the problem of shadow removal from multi-projector displays. Their fundamental contribution was the use of more than one projector to cover the same area of the display surface. With this redundant configuration, when an occlusion occurs in front of one projector, the content of the projection remains visible on the display surface. To lower the chances that more than one projector become occluded simultaneously, they are placed as far apart as possible. In this manner, although the content remains visible, *partial shadows* appear when the light from only one projector is occluded. To compensate, shadows are detected on the display surface, and the brightness of all projectors in that region is increased, as described in the following paragraph.

The design of this first shadow removal system uses a camera that is oriented to capture images from the display surface. Since the system requires an unoccluded view, the camera is placed as close as possible to the surface, ideally attached to the ceiling. The user can choose the area of display by selecting the four desired corners in the camera image, or the system can automatically detect brighter areas in the image and construct a quadrilateral that it will use as display area. Homographies between the camera and the projectors are computed from this data. Image rectification is accomplished using homographies in a manner similar to the method we use in our work, first described by Raskar *et al.* [30, 31, 32]. To initialize the shadow elimination, the system takes a few frames of the unoccluded display surface and creates a statistical model for each pixel in a manner similar to background subtraction techniques. After that all images from the cameras are compared pixel-by-pixel to the model. To compensate for deviations, one alpha mask[1] shared by all projectors is generated. With this mask, we can increase the intensity of pixels that require more light, and dim pixels from

[1] An alpha mask is an image where the value of each pixel indicates the desired brightness.

previously shadowed areas that are now too bright. This simple procedure does not need any illumination model and works correctly for a static image, but not for dynamic video, and it also requires at least three frames to converge to the correct brightness. Moreover, the method actually increases illumination on occluding objects, which is unpleasant for users who are facing the projectors.

In parallel, Jaynes *et al.* [23] developed a similar approach, but instead of assuming a static image, the system uses a basic radiometric model to predict the expected colors in the camera view. This way one can change the image being projected without needing to reinitialize the system after each update. The model consists of a color transfer function that the system estimates through an automated calibration process that iteratively projects different color intensities uniformly across the display surface, and for each intensity measures the response of the camera, where the three color channels (red, green, and blue) are calibrated separately. Similarly to the previous approach, this method also increases illumination on occluding objects and requires three or four frames to converge to the correct brightness. In later work [22], the authors note that this still does not permit operation at 60 Hz, the threshold at which human perception of flickering is significantly reduced [15]. One of our goals is to achieve reasonable performance at such frame rates, without increasing illumination on occluding objects.

Cham *et al.* [6] proposed a method to prevent such unnecessary and distracting illumination of occluding objects. This employs a probing technique involving more than one alpha mask, one for each projector. When the system detects reduced pixel brightness in the camera image, it appropriately modulates the pixel intensity, one projector at a time. The projector that has no influence on the intensity as seen by the camera is the occluded projector. This probing technique runs continuously and iteratively, and takes on average five to seven frames before converging. They also tested a binary approach where each pixel is either on or off. Although they describe this switching process as “extremely fast”, it actually runs at three frames per second. Flagg *et al.* [14] optimized this system and obtained a performance of eight frames per second, but this is still insufficient for smooth video playback.

Jaynes *et al.* [22] continued perfecting their method to achieve better results for dynamic interactive applications, where shadow removal needs to work with a constantly changing display. They enhanced their initial work [23] by adding the probing technique of Cham *et al.* [6]. Also, they added support for more than one camera, in case one or more cameras become occluded and cannot see the display surface anymore. In addition, they proposed a technique that computes the desirability of a projector for a given region of the display surface. The desirability is proportional to the resolution the projector can achieve in this region, taking into account warping induced by projecting at an angle. The system then attempts to fill regions of the display surface with the more desirable projectors, and only uses less desirable ones in case of occlusion. Again, the system can either work on individual pixels or approximate the shadow removal to rectangular regions, boosting the processing performance from two to nine frames per second. However, the method still requires three or four frames to converge to the correct brightness adjustment. Hilario and Cooperstock [17, 18] developed similar techniques as well with comparable results.

Despite all the research done on shadow detection, by design, these methods can only remove shadows after they appear, which is their most important limitation. Moreover, it is not always possible to remove shadows after a single frame. Convergence can often only be attained after a few frames. This is because the system needs to make assumptions about whether or not a projector is occluded and then needs to test these assumptions on the display surface, producing a feedback loop between the projectors and the cameras. If the system incorrectly switches pixels with the full intensity after only one frame, visually distracting echo may result. Obviously, this comparison operation between the cameras and the projectors must also be time synchronized. For this reason, implementing such methods becomes more complex in the case of video at high frame rates since to compare correctly we need to know the precise delay incurred in the projectors, in the cameras, and in processing. An additional constraint is that together the cameras need an unoccluded view of the entire display surface, or the system cannot function properly.

1.3.2 Occlusion Detection Methods

An alternative approach to shadow detection is occlusion detection, where instead of analyzing the display surface to detect shadowed regions, the system detects the occlusions themselves in front of the projectors. Much research has gone into detecting occlusion in front of color camera and projector pairs to localize hands for human-computer interaction applications, but none of these techniques has been applied to shadow removal. Licsar and Sziranyi [25] use a method that needs a geometric and color calibration similar to the ones used by shadow detection methods [17, 18, 22, 23], and as such would produce comparable results as these if applied to shadow removal. Pinhanez *et al.* [27] use frame differencing, taking the difference between two consecutive camera frames, but this can only detect moving objects. It would not work for people standing still. Takao *et al.* [41] limits the interaction outside the projection area, such that projector occlusion or a shadow never occurs. Von Hardenberg [46] uses a running average to adapt the model of the background, which is the display surface, but this technique can only adapt to slowly changing content. In brief, none of these methods using color cameras has the potential to offer adequate occlusion detection required to remove shadows from a display surface with video running at 60 frames per second.

Another way of dealing with this problem using color cameras would be to use *imperceptible patterns* that were first described by Raskar *et al.* [31] and further enhanced by Cotting *et al.* [9]. With this method the projectors display patterns to generate a texture, such as binary stripes typically used in structured light approaches or coded tags typically used in augmented reality applications. The cameras can then detect the regions with missing texture, which correspond to where occlusion occurs. It is rendered imperceptible to the human eye [15] by rapidly switching between complementary patterns at 60 Hz. Capturing them however requires a synchronized camera. We believe that this approach could effectively be applied to shadow removal, but no available publication describes such an application. In any case, the drawback is that we need cameras and projectors that can be synchronized with one another, specialized hardware that can be expensive.

Instead of color cameras, one can use infrared cameras, which are insensitive to light emitted by projectors. The system by Sato *et al.* [33] uses a thermal infrared camera to detect occlusions by hands. From the camera image, the occlusions are directly and clearly visible with no further processing required, although a drawback is the high cost of thermal cameras, typically an order of magnitude more expensive than color or near infrared cameras.

Other methods use a near infrared camera mounted alongside each projector [13, 42] to detect the occlusion. Because these cameras do not see the projector light, a simple background subtraction technique can be used to generate a pixel-mask of the occlusions in front of each camera-projector pair. Since background subtraction requires good contrast between the background (display surface) and the foreground (people), infrared floodlights are used to illuminate the display surface, while not illuminating the people. This method recovers occlusion masks of high quality, which are directly used to generate alpha masks for the projectors. Furthermore, Flagg *et al.* [13] perform much of their computation on Graphics Processing Units (GPUs), which greatly improves processing speed and provides a total latency of 53 ms. The system is fast enough to follow typical human gestures they tested. The main drawback of this approach is the requirement of hardware specifically for shadow removal purposes, which may include infrared floodlights, and for each projector, a near infrared camera and additional processing power (GPUs).

Summet *et al.* also conducted usability experiments [35, 36] and a thorough technical review [37] of all the shadow removal methods described in this section. Their research shows that although users preferred back projection to front projection with shadow removal, they still preferred front projection with shadow removal over front projection without shadow removal. However, as explained earlier in this section, rear projection requires more space, shadow detection methods do not cope well with dynamic video projection, and occlusion detection methods that work better all require the installation of hardware that can be expensive or cumbersome. Therefore, this justifies our development of a new method.

1.4 Object Tracking for Shadow Removal

As an alternative to the shadow detection and near infrared occlusion detection methods, we propose object tracking as the basis of our approach to shadow removal [2]. We were attracted to object tracking because many applications already use it, for example to change video or audio according to the position of people, as envisioned by Cruz-Neira *et al.* [11, 12], to ensure that images are rendered from the proper perspective [31] or to adjust 3D audio according to user movement [50]. Using tracking for shadow removal thus leverages hardware and technology that may already be in place, since our method can use data from many typical object trackers, reducing the complexity and cost of installation.

Also, tracking can take advantage of temporal information to predict the motion of people as occluders.

Our current approach requires a manual calibration method for both the cameras and projectors, an adequately reliable object tracker, and an algorithm that combines this information to perform shadow removal. While calibration and tracking are often inaccurate and imprecise, we show that good results can be obtained from our system.

1.5 Research Overview

A high-level diagram of our system architecture appears in Figure 1–2. Our work focuses on the Calibration, Object Tracking, Shadow Removal, and Image Rectification modules. We did not address issues of color correction and multi-focal projection. In the following chapters, we elaborate on our approach.

Chapter 2 first describes in detail the camera calibration procedure we employed, and how we adapted it to projector calibration. To find the intrinsic parameters of our cameras, we capture a checkerboard pattern from many different angles, extracting the corners of the checkerboard, and performing, in accordance with the theory of multiple view geometry, mathematical operations on the locations of these extracted points to find the internal parameters or *intrinsics* of the camera. For the projectors, since they cannot capture images, for the duration of the calibration process, each projector is paired with a camera. To find the intrinsics of the projector, images from the scene are captured by the camera while a checkerboard pattern is displayed by the projector on another checkerboard pattern printed on a board. We can then use a procedure almost

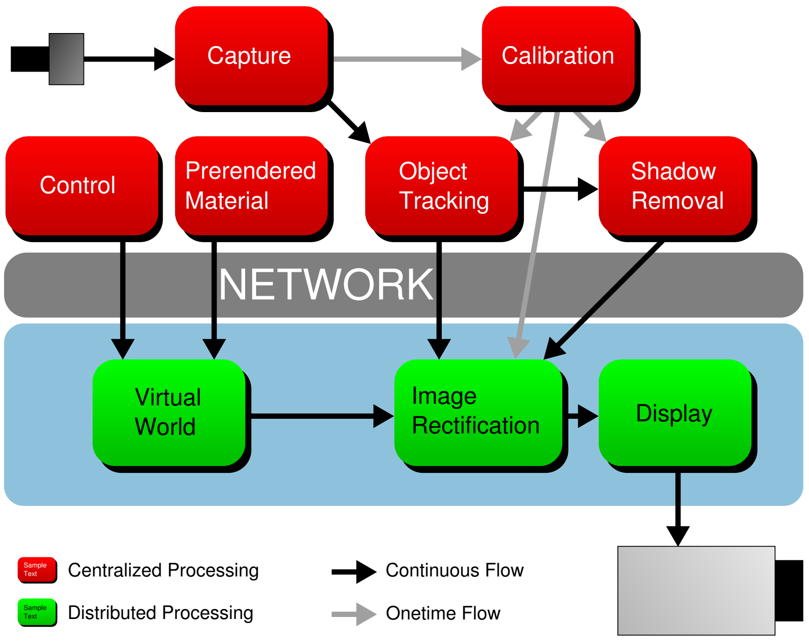


Figure 1–2: Illustration of the flow of data in our system.

identical to the one used for the cameras. To find the *extrinsics*, the external parameters corresponding to the orientations and positions in space of the cameras and projectors, we devised a method that uses the display surface, and for which we do not need to manipulate an object. With this method we can also recover the planes of the display surface, the floor, the ceiling, the walls, and of other flat surfaces, which we use as a 3D model of our environment for the following modules.

Next, Chapter 3 describes how we adapted a camera-based object tracker for our purposes. The main challenge we faced was to achieve reasonably reliable tracking, despite the projection of dynamic video in the background, which can easily be misinterpreted as moving people. To render the tracker insensitive to video projection, we decided to use two color cameras in concert with the calibration data and geometry of the environment modeled as planes to generate a *disparity map*. This provides the pixel correspondence between the images of two cameras. Taking the difference of the two images using this disparity map produces *disparity contours* for objects that are not part of our model, such as people. Objects that are part of the model, such as the display surface, will not appear in the disparity contours, despite the video projection, thus achieving our goal. We can then use the disparity contours as input images to a blob tracker to obtain tracking information of people in the scene.

Chapter 4 explains how we integrated these components to model shadows and achieve shadow removal. The model represents tracked people in 3D space as rectangles parallel to the display surface. Using the projector calibration information, we can back-project the rectangles onto the display surface to find regions on the display surface where shadows are occurring, according to our model. Coverage masks are generated from this information, which are then processed using the distance transform for intensity blending and finally normalized. Before display, the masks undergo gamma correction and image rectification in OpenGL.

Chapter 5 summarizes our experimental results. We achieved an accurate calibration of the intrinsic parameters, although the extrinsic parameters recovered by our simple method were not as accurate. This however did not affect the qualitative results, since increasing by only 20% the dimensions of tracking rectangles produced correct behavior in the case of static images. For dynamic video projections, the tracking method using disparity contours often failed due to inconsistencies between the color responses of the two cameras, but the results are nevertheless promising.

Chapter 6 concludes with a discussion of limitations and future work. Currently, the tracking module cannot cope well with video projection, the manual calibration process is tedious, and the shadow removal method only works with flat display surfaces. We provide design ideas for a better method that has a more user friendly calibration process and that works with arbitrarily shaped display surfaces. Nevertheless, the current framework has the advantage of offering more opportunities for further development. For example, knowing the position in space of display surfaces, the system can automatically place corresponding cameras in a virtual world to provide to a user the correct perspective.

CHAPTER 2
Camera and Projector Calibration

In the first section of this chapter, we describe the basics in multiple view geometry that are required to understand the camera calibration procedure. Hartley and Zisserman [16] provides a more exhaustive explanation of these concepts. The camera calibration requires many images of a checkerboard pattern captured from different orientations, whose corners are then extracted, and used eventually to find the intrinsics or internal parameters of the camera. We can think of the projector as the dual of the camera, so we can calibrate it using the same theory. However, since it cannot capture images, it needs to be calibrated indirectly through images captured by a camera viewing a projected checkerboard pattern. Finally, since our goal is to build a 3D model of the environment or room where video projection takes place, we need to find the extrinsics, which are the orientations and positions in space of the cameras and projectors. We also need to find the planes of the display surface, of the floor, and of any other flat surfaces required for the 3D model.

2.1 Fundamentals in 3D Geometry

Before delving further into calibration procedures, we first provide an introduction to multiple view geometry in 3D. The most fundamental concept, a point, can be represented in 3D space by a column vector with three components $\mathbf{X} = [\ x\ y\ z\ ]^{\mathrm{T}}$, also know as Cartesian coordinates. Since it has three parameters

x, y, and z, we say that this entity has three *degrees of freedom.* However, this representation does not allow easy computation of projective transformations using linear algebra and matrix notation, as described later in this section. If we augment the 3-vector with one more component $w \neq 0$, the vector becomes $\mathbf{X} = [\ x'\ y'\ z'\ w\ ]^{\mathrm{T}}$ where $x' = wx$, $y' = wy$, $z' = wz$, and we can then represent any projective transformations using matrices. This 4-vector representation is known as *homogeneous coordinates.* The component w is the nonzero scale of the vector, which is arbitrary, so the vector still has three degrees of freedom, not four. When a vector is normalized such that $w = 1$, the first three components of the homogeneous vector correspond to the Cartesian coordinates of the 3D point in Euclidean space.

Next, a 3D point $\mathbf{X} = [\ x\ y\ z\ ]^{\mathrm{T}}$ lies on a plane if and only if the following equation is satisfied: $ax + by + cz + d = 0$ where the constants a, b, c, and d represent the plane. This equation can be extended in the case of homogeneous coordinates by multiplying it by the scale w: $ax' + by' + cz' + dw = 0$. Given the row vector $\mathbf{p} = [\ a\ b\ c\ d\ ]$ and a point $\mathbf{X}$ expressed in homogeneous coordinates, we can also represent the equation of a plane more compactly using linear algebra: $\mathbf{pX} = 0$. Note that $\mathbf{n} = [\ a\ b\ c\ ]^{\mathrm{T}}$ is a *normal vector* of the plane, a vector perpendicular to it. Also, with homogeneous coordinates, it is possible to have points with components $w = 0$ if at least one of the other components is nonzero. Such a vector indicates a *point at infinity* and lies on *the plane at infinity.* Since we only require $w = 0$ for such points, we must have $a = b = c = 0$ and $d \neq 0$, which is typically represented by the plane vector $\mathbf{p}_\infty = [\ 0\ 0\ 0\ 1\ ]$. A point at infinity is also a *directional vector*. Since it is also only defined up to scale, any vector in the same direction also represents the same point at infinity. Also, when added to a normal finite point, the scale of the resulting vector is the same as the initial finite point, and the operation is equivalent to a simple vector addition in Cartesian space, *e.g.*:

$$[\ x_1\ y_1\ z_1\ w\ ]^{\mathrm{T}} + [\ x_2\ y_2\ z_2\ 0\ ]^{\mathrm{T}} = [\ x_1+x_2\ \ y_1+y_2\ \ z_1+z_2\ \ w\ ]^{\mathrm{T}}. \tag{2.1}$$

In the next section, we detail some important entities that live on planes and how they can be operated upon.

2.1.1 Geometry on Planes

On a plane, we can define lines, conics, and projective transformations. Let our plane be where $z = 0$. Since we can adjust the world coordinate frame such that every plane lies at $z = 0$, the following discussion is valid for every plane in 3D space, but we will denote them collectively as *the scene plane* for clarity. We can thus drop the z component from the notation when dealing with 2D entities in 3D space.

A *line* on the scene plane is defined similarly to a plane in 3D space. A homogeneous 2D point $\mathbf{x} = [\ x'\ y'\ w\ ]^{\mathrm{T}}$ lies on the line $\mathbf{l} = [\ a\ b\ d\ ]$ if and only if $\mathbf{lx} = 0$. The scene plane also has a *line at infinity* $\mathbf{l}_\infty = [\ 0\ 0\ 1\ ]$. In other words, since $z = 0$, the line lies on the scene plane, and since for all points on the line we have $w = 0$, it lies on the plane at infinity as well. This shows that any plane $\mathbf{p}$ intersects the plane at infinity at a line, and that this line is the line at infinity of the plane $\mathbf{p}$.

We can also define *conics* on the scene plane. A point $\mathbf{x} = [\ x\ y\ ]^{\mathrm{T}}$ lies on a conic if and only if the following equation is satisfied: $ax^2+bxy+cy^2+dx+ey+f = 0$. The non-degenerate conics are the circle, the ellipse, the parabola, and the hyperbola. Again, we can augment the equation with homogeneous coordinates:

$$ax'^2 + bx'y' + cy'^2 + dwx' + ewy' + fw^2 = 0 \tag{2.2}$$

or more simply reformulated using matrix notation:

$$\mathbf{x}^{\mathrm{T}}\mathrm{C}\mathbf{x} = \mathbf{x}^{\mathrm{T}} \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \mathbf{x} = 0 \tag{2.3}$$

Since this equation makes use of homogeneous coordinates, the conic matrix C is a *homogeneous matrix* and is only defined up to scale. Therefore, it actually has five degrees of freedom, not six.

A point on the scene plane can also undergo *linear transformation or distortion* via a *homography*, which is also known as a *projective transformation*. This transformation can be represented by an invertible 3×3 matrix H, which can transform a homogeneous point $\mathbf{x}$ into $\mathbf{x}' = \mathrm{H}\mathbf{x}$. Since this operation works on homogeneous coordinates, the matrix H is again a homogeneous matrix. Consequently, it only has eight degrees of freedom, not nine. Lines and conics can also be transformed as follows:

$$\mathbf{l}' = \mathbf{l}\,\mathrm{H}^{-1}, \tag{2.4}$$

$$\mathrm{C}' = \mathrm{H}^{-\mathrm{T}}\,\mathrm{C}\,\mathrm{H}^{-1}. \tag{2.5}$$

Lines and conics are transformed in this manner so that if $\mathbf{x}$ lies on $\mathbf{l}$ and C, then $\mathbf{x}'$ lies on $\mathbf{l}'$ and C$'$, in other words:

$$\mathbf{l}'\mathbf{x}' = \mathbf{l}\,\mathrm{H}^{-1}\mathrm{H}\mathbf{x} = \mathbf{l}\mathbf{x} = 0, \tag{2.6}$$

$$\mathbf{x}'^{\mathrm{T}}\,\mathrm{C}'\,\mathbf{x}' = \mathbf{x}^{\mathrm{T}}\mathrm{H}^{\mathrm{T}}\mathrm{H}^{-\mathrm{T}}\mathrm{C}\,\mathrm{H}^{-1}\mathrm{H}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathrm{C}\mathbf{x} = 0. \tag{2.7}$$

In addition to operations solely limited to a 2D world, it is possible to project points from a 3D space onto a plane, the image plane, as detailed next.

2.1.2 Projection onto the Image Plane

Using homogeneous coordinates, we can denote the *projection* of a 3D point $\mathbf{X}$ onto the 2D *image plane* as:

$$\mathbf{x} = \mathrm{P}\mathbf{X}, \tag{2.8}$$

where P is a 3×4 projection matrix, and $\mathbf{x}$ a 2D homogeneous vector. Since we are working with homogeneous coordinates, P is again a homogeneous matrix that is defined only up to scale. Thus it only has eleven degrees of freedom, not twelve. We can decompose the matrix in the following way:

$$\mathrm{P}_{3\times4} = \mathrm{M}_{3\times3}[\mathrm{I}_{3\times3}|-\mathbf{C}] = \mathrm{K}_{3\times3}\mathrm{R}_{3\times3}[\mathrm{I}_{3\times3}|-\mathbf{C}] = \lambda \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \mathrm{R}[\,\mathrm{I} \mid -\mathbf{C}\,] \tag{2.9}$$

where λ is an arbitrary scaling factor, K is the upper triangular camera matrix (composed of the *intrinsics*: the focal length $[\ f_x\ f_y\ ]^{\mathrm{T}}$, the principal point $[\ x_0\ y_0\ ]^{\mathrm{T}}$ and the skew factor s), R is the orthogonal rotation matrix, and $\mathbf{C}$ is the camera center (translation). The decomposition of M into KR is known as the

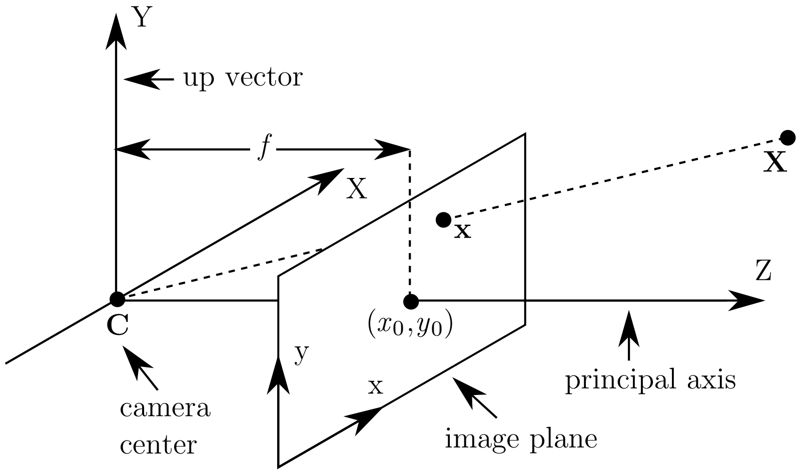


Figure 2–1: Illustration of the pinhole camera model. The line joining the camera center **C** to the 3D point **X** intersects the image plane at the 2D point **x**, which we call the projection of point **X**. In this ideal case, the focal length $f_x = f_y = f$ and the skew $s = 0$. This simple illustration also assumes a camera center at the origin with no rotation.

RQ factorization [16], a close relative to the more commonly known *QR factorization* [28], widely known for its existence and uniqueness, when M has full rank. The matrix K has five parameters up to an arbitrary scale λ, and there are three rotation axes and three translation components, so the projection has a total of eleven degrees of freedom as previously noted.

This projection is commonly used as model for the *pinhole camera*, as illustrated in Figure 2–1. The projection accounts for all linear distortions or transformations that take place when a pinhole camera projects an image. Most cameras using lenses follow this model, but also introduce non-linear radial and tangential distortions. These have to be taken into account during calibration, as we describe later in this chapter. Before doing so, we explain the projection of the scene plane using the simple linear model.

In the general case, points lying on the scene plane in 3D space will fill the image plane of the camera, undergoing a projective transformation. Let us consider once more the scene plane at $z = 0$. We can then determine the correspondence between 2D points $\mathbf{x}$ on the image plane and 3D points $\mathbf{X}$ on the scene plane as follows:

$$\mathbf{x} = \mathrm{P}\mathbf{X} = [\ \mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_3\ \mathbf{p}_4\ ][\ x\ y\ 0\ 1\ ]^{\mathrm{T}} = [\ \mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_4\ ][\ x\ y\ 1\ ]^{\mathrm{T}} \tag{2.10}$$

where the matrix $[\ \mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_4\ ]$ is simply a homography, or a projective transformation. Moreover, this homography contains information about intrinsics K, the rotation R and the camera center $\mathbf{C}$:

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{KR}[\ \mathrm{I} \mid -\mathbf{C}\ ][\ x\ y\ 0\ 1\ ]^{\mathrm{T}} = \mathrm{KR}[\ \mathbf{e}_1\ \mathbf{e}_2\ -\mathbf{C}_i\ ][\ x\ y\ 1\ ]^{\mathrm{T}}. \tag{2.11}$$

At infinity, the resulting homography is invariant to translation. Points lying on the plane at infinity $\mathbf{X}_\infty = [\ x\ y\ z\ 0\ ]$ are projected as

$$\mathbf{x} = \mathrm{P}\mathbf{X}_\infty = \mathrm{KR}[\ \mathrm{I} \mid -\mathbf{C}\ ][\ x\ y\ z\ 0\ ]^{\mathrm{T}} = \mathrm{KR}[\ x\ y\ z\ ]^{\mathrm{T}} \tag{2.12}$$

such that the homography $\mathrm{H} = \mathrm{KR}$, which is therefore invariant to translation. We can intuitively understand this by looking at the stars at night and noting that they do not move when we do.

2.1.3 Back-Projection from the Image Plane

In addition to projecting onto the image plane, it is also possible to *back-project* a point $\mathbf{x}$ from the image plane and find a 3D point $\mathbf{X}$, which projects to this point $\mathbf{x}$, in other words

$$\mathbf{X} = \mathrm{B}\mathbf{x} \tag{2.13}$$

where B is the *back-projection matrix* such that $\mathbf{x} = \mathrm{P}\mathbf{X}$. We also want the point $\mathbf{X}$ to lie on a plane $\mathbf{p}$, as described next.

A point

$$\mathbf{D} = \begin{bmatrix} \mathrm{M}^{-1} \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{x} \tag{2.14}$$

is a 3D point at infinity ($w = 0$) that correctly projects to the point $\mathbf{x}$ on the image plane since

$$\mathrm{P}\mathbf{D} = [\ \mathrm{M} \mid \mathbf{p}_4\ ] \begin{bmatrix} \mathrm{M}^{-1} \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{x} = \mathbf{x} \tag{2.15}$$

where M is as defined in Equation 2.9.

As described previously by Equation 2.1, a point at infinity is also a directional vector, which we can use to travel along a line. The line that passes through the point $\mathbf{x}$ on the image plane and the 3D point $\mathbf{X}$ also goes through the center of the camera $\mathbf{C}$, as shown in Figure 2–1, so by using $\mathbf{D}$ we can back-project the point $\mathbf{x}$ into

$$\mathbf{X} = \mathbf{C} + t\mathbf{D}. \tag{2.16}$$

We also want the point $\mathbf{X}$ to lie on the plane $\mathbf{p}$, such that $\mathbf{pX} = 0$ or by substitution $\mathbf{p}(\mathbf{C} + t\mathbf{D}) = 0$. By isolating t and resubstituting in Equation 2.16, we obtain

$$\mathbf{X} = \mathbf{C} - \left(\frac{\mathbf{pC}}{\mathbf{pD}}\right)\mathbf{D}\ , \tag{2.17}$$

which can be rearranged to obtain Equation 2.13 where

$$\mathrm{B} = [\ \mathbf{C}\,\mathbf{p} - \mathbf{p}\,\mathbf{C}\,\mathrm{I}_{4x4}\ ] \begin{bmatrix} \mathrm{M}^{-1} \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{2.18}$$

Using this fundamental knowledge of geometry, the next section introduces concepts required to understand the calibration procedure for cameras and projectors explained later in this chapter.

2.2 Calibration from Planes

The calibration consists of finding the intrinsic and extrinsic parameters of both the cameras and the projectors. These parameters are required to build the 3D model of the room, which is then needed to find the location and size of people in 3D and to model their shadows occurring on the display surface. In this section, we describe how to calibrate cameras and projectors using only the scene plane.

2.2.1 Intrinsic Parameters

The intrinsic parameters that we want to recover for the cameras include their matrices K and any nonlinear distortion parameters. The calibration method we describe here was first developed by Zhang [51] and later implemented and refined by Bouguet [5]. This is currently a popular calibration method as can be seen from the large number of references to the article in the literature. It does not require complex hardware installations as previous approaches. Before describing the steps involved in this method, which are corner extraction, estimation of homographies, and intrinsic and extrinsic parameters, we first explain two special geometric entities: the absolute conic and the circular points.

2.2.2 A Special Conic: The Absolute Conic

The *absolute conic* Ω_∞ is defined with the conic matrix $\mathrm{C} = \mathrm{I}$, the identity matrix, for points that lie on the plane at infinity $\mathbf{p}_\infty = [\ 0\ 0\ 0\ 1\ ]$. In other words, a point $\mathbf{X} = [\ x\ y\ z\ w\ ]$ lies on the absolute conic if and only if

$$x^2 + y^2 + z^2 = 0, \tag{2.19}$$

$$w = 0. \tag{2.20}$$

This conic contains no real points, only imaginary ones. However, it is useful for calibration purposes. We see from Equation 2.12 that the projection of points from the plane at infinity to the image plane induces the homography $\mathrm{H} = \mathrm{KR}$. Using Equations 2.5 and 2.12 to project a conic from the plane at infinity, we find that the absolute conic appears on the image plane of a camera as

$$\begin{aligned} \omega &= \mathrm{H}^{-\mathrm{T}}\Omega_\infty\mathrm{H}^{-1} = \mathrm{H}^{-\mathrm{T}}\mathrm{I}\ \mathrm{H}^{-1} = (\mathrm{KR})^{-\mathrm{T}}(\mathrm{KR})^{-1} = \mathrm{K}^{-\mathrm{T}}\mathrm{R}^{-\mathrm{T}}\mathrm{R}^{-1}\mathrm{K}^{-1} \\ \omega &= (\mathrm{KK}^{\mathrm{T}})^{-1}. \end{aligned} \tag{2.21}$$

We can then decompose ω^{-1} by Cholesky factorization [16, 28] and extract K, which is the camera matrix that contains the intrinsics. Note that ω is invariant to rotation and translation as the matrix R is factored out during the projection, and a projection from the plane at infinity is invariant to translation. We can use this fact to our advantage by capturing the absolute conic from many different angles and positions, whose image will always be ω for a given camera with intrinsics K.

Unfortunately, since the absolute conic is imaginary, we cannot capture it using a real camera. Using real points, we can however find constraints and transfer them onto the image of the absolute conic, using what are called the circular points.

2.2.3 The Circular Points

The *circular points* are the complex conjugate imaginary points at infinity

$$c_+ = \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix}, \quad c_- = \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix} \tag{2.22}$$

where the last coordinate is $w = 0$, but could also be $z = 0$ when considering the scene plane. They are special points since they lie both on the line at infinity of the scene plane:

$$\mathbf{l}_\infty \mathbf{c}_+ = [\ 0\ 0\ 1\ ][\ 1 \quad i\ 0\ ]^{\mathrm{T}} = 0, \tag{2.23}$$

$$\mathbf{l}_\infty \mathbf{c}_- = [\ 0\ 0\ 1\ ][\ 1\ -i\ 0\ ]^{\mathrm{T}} = 0, \tag{2.24}$$

as well as on the absolute conic on the plane at infinity:

$$\mathbf{c}_+{}^{\mathrm{T}} \Omega_\infty \mathbf{c}_+ = [\ 1 \quad i\ 0\ ]\ \mathrm{I}\ [1 \quad i\ 0\ ]^{\mathrm{T}} = 1 - 1 = 0, \tag{2.25}$$

$$\mathbf{c}_-{}^{\mathrm{T}} \Omega_\infty \mathbf{c}_- = [\ 1\ -i\ 0\ ]\ \mathrm{I}\ [1\ -i\ 0\ ]^{\mathrm{T}} = 1 - 1 = 0. \tag{2.26}$$

Their name, "circular points", comes from the fact that they lie on all circles of the scene plane. To show this, a point lies on a circle if it satisfies Equation 2.2 with $a = c$ and $b = 0$, equating to $ax^2 + ay^2 + dwx + ewy + fw^2 = 0$. For points at infinity where $w = 0$, this reduces to $ax^2 + ay^2 = 0$, and for the circular points $a - a = 0$.

We can also transform these points by homography just as any other points:

$$\mathbf{c}'_+ = \mathrm{H}\mathbf{c}_+, \quad \mathbf{c}'_- = \mathrm{H}\mathbf{c}_-. \tag{2.27}$$

If we let this homography be the projection onto the camera image plane according to Equation 2.12, the two imaged points lie on the image of the absolute conic ω:

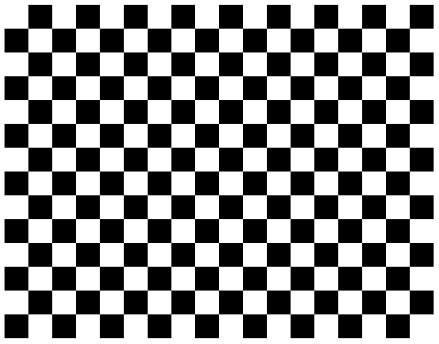
$$(\mathrm{H}\mathbf{c}_\pm)^\mathrm{T}\omega(\mathrm{H}\mathbf{c}_\pm) \;=\; \mathbf{c}_\pm{}^\mathrm{T}\mathrm{H}^\mathrm{T}\mathrm{H}^{-\mathrm{T}}\Omega_\infty\mathrm{H}^{-1}\mathrm{H}\mathbf{c}_\pm \;=\; \mathbf{c}_\pm{}^\mathrm{T}\Omega_\infty\mathbf{c}_\pm = 0 \tag{2.28}$$

which is true as shown above.

As Equation 2.3 tells us, knowing five points on ω, the information we are seeking about our camera can be recovered completely. According to the discussion above, thanks to the circular points, we can do this by using three homographies H induced by the scene plane in the real world. Before we can find those however, we must first be able to measure or extract real points from the scene plane as observed by the camera. This brings us to the problem of finding points on a real plane using corner extraction.

2.2.4 Corner Extraction

We place a *calibration board* containing a typical checkerboard pattern as shown in Figure 2–2(a) onto the scene plane to find point correspondences between it and the image plane of the camera. The corners of the pattern are the points $\mathbf{x}_i$ on the scene plane. They are known in advance and decided upon printing of the board. These corners are imaged by the camera on the image plane as shown in Figure 2–2(b), points that we denote as $\mathbf{x}'_i$. To extract these imaged points with sub-pixel precision, we use the following procedure.


(a) Calibration pattern.

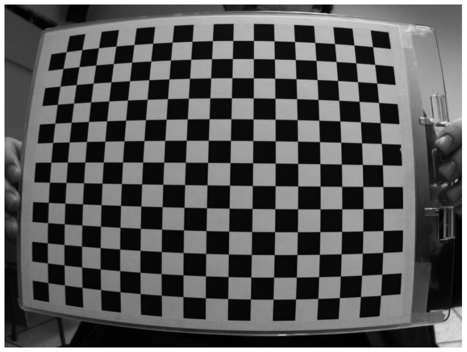

(b) Sample image from camera.

Figure 2–2: The calibration pattern that we use, and a sample image from the camera when holding the calibration board in front.

In the ideal case as depicted in Figure 2–3, a center pixel $\mathbf{x'}_i$ is also a *corner pixel*. For all pixels $\mathbf{x'}_j$ in the neighborhood of this $\mathbf{x'}_i$ we either have that their gradients $\nabla_{\mathbf{x'}_j} = \mathbf{0}$ or that $\nabla_{\mathbf{x'}_j}$ is perpendicular to the vector $\mathbf{x'}_i - \mathbf{x'}_j$. Both constraints can be expressed by the equation

$$\nabla^{\mathrm{T}}_{\mathbf{x'}_j}(\mathbf{x'}_i - \mathbf{x'}_j) = 0. \tag{2.29}$$

Due to noise and other sources of error, this will never be exactly true, but under the assumption of Gaussian noise, we can nonetheless use this as a constraint to find the maximum likelihood estimate for a given $\mathbf{x'}_i$. As a neighborhood, we use a typical square window of 11×11 pixels centered at $\mathbf{x'}_i$. Using an initial estimate provided by the user, the constraint imposed by the sum

$$\sum_{j \in \text{window}} \nabla^{\mathrm{T}}_{\mathbf{x'}_j}(\mathbf{x'}_i - \mathbf{x'}_j) = 0 \tag{2.30}$$

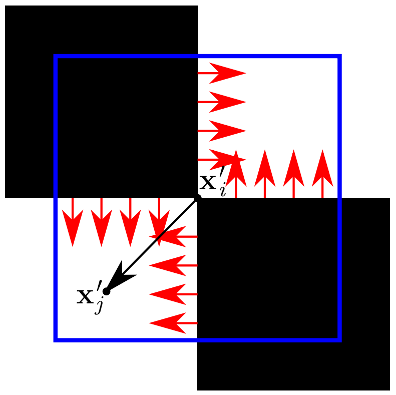


Figure 2–3: Ideal corner $\mathbf{x'}_i$ at the center of the sample neighborhood window depicted with the blue frame. Only points on edges have gradients larger than $\mathbf{0}$ as illustrated with the red arrows.

over all pixels $\mathbf{x'}_j$ in the window can be reformulated using the *gradient covariance matrix* $\mathrm{G}_j = \nabla_{\mathbf{x'}_j}\nabla^{\mathrm{T}}_{\mathbf{x'}_j}$:

$$\begin{aligned}\sum_j \mathrm{G}_j(\mathbf{x'}_i - \mathbf{x'}_j) &= \mathbf{0},\\ (\sum_j \mathrm{G}_j)\mathbf{x'}_i - (\sum_j \mathrm{G}_j\mathbf{x'}_j) &= \mathbf{0},\\ (\sum_j \mathrm{G}_j)\mathbf{x'}_i &= (\sum_j \mathrm{G}_j\mathbf{x'}_j),\end{aligned} \tag{2.31}$$

which can be seen as a linear system of the form $\mathrm{A}\mathbf{x} = \mathbf{b}$. Solving for $\mathbf{x'}_i$ generates a new estimate closer to the corner. The algorithm runs iteratively and stops when the movement of the center no longer exceeds some threshold ϵ, typically 0.005 pixels. According to Bouguet [5], this procedure can extract corners with an accuracy of up to 0.1 pixels, although no proof of convergence is provided. With the two sets of points $\mathbf{x}_i$ and $\mathbf{x'}_i$, we can then compute an estimate of the homography induced by the scene plane.

2.2.5 Homography Estimation

We can estimate the homography induced by the projection of the scene plane onto the image plane of a camera using the *direct linear transformation* (DLT) algorithm described in this section. This algorithm requires four or more point correspondences between the two planes, which we find using the corner extraction procedure described above.

Given a set of n 2D points $\mathbf{x}_i$ on the scene plane where $i = 1, ..., n$ and their correspondences $\mathbf{x'}_i$ on the image plane, we are looking for a homography H such that

$$\mathbf{x'}_i = \lambda_i \mathrm{H}\mathbf{x}_i \tag{2.32}$$

for all $i = 1, ..., n$ where λ_i are arbitrary scaling factors, since we are working with homogeneous coordinates. We can factor out λ_i by working instead with the vector cross product

$$\mathbf{x'}_i \times \mathrm{H}\mathbf{x}_i = \mathbf{0}. \tag{2.33}$$

Since the scale does not change the direction of $\mathrm{H}\mathbf{x}_i$, we still obtain $\mathbf{0}$ regardless of its value. If we denote the homography $\mathrm{H} = [\ \mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3\ ]^{\mathrm{T}}$ by its row vectors, then

$$\mathrm{H}\mathbf{x}_i = [\ \mathbf{h}_1\mathbf{x}_i\ \mathbf{h}_2\mathbf{x}_i\ \mathbf{h}_3\mathbf{x}_i\ ], \tag{2.34}$$

and we can expand the vector cross product as

$$\mathbf{x'}_i \times \mathrm{H}\mathbf{x}_i = \begin{bmatrix} y'_i\mathbf{h}_3\mathbf{x}_i - w'_i\mathbf{h}_2\mathbf{x}_i \\ w'_i\mathbf{h}_1\mathbf{x}_i - x'_i\mathbf{h}_2\mathbf{x}_i \\ x'_i\mathbf{h}_2\mathbf{x}_i - y'_i\mathbf{h}_1\mathbf{x}_i \end{bmatrix} = \mathbf{0}. \tag{2.35}$$

This provides three equations equal to zero. We can rewrite them as a linear system in the usual A$\mathbf{x} = \mathbf{b}$ form

$$\begin{bmatrix} & -w_i'\mathbf{x}_i^{\mathrm{T}} & y_i'\mathbf{x}_i^{\mathrm{T}} \\ w_i'\mathbf{x}_i^{\mathrm{T}} & & -x_i'\mathbf{x}_i^{\mathrm{T}} \\ -y_i'\mathbf{x}_i^{\mathrm{T}} & x_i'\mathbf{x}_i^{\mathrm{T}} & \end{bmatrix} \begin{bmatrix} \mathbf{h}_1^{\mathrm{T}} \\ \mathbf{h}_2^{\mathrm{T}} \\ \mathbf{h}_3^{\mathrm{T}} \end{bmatrix} = \mathbf{0}. \tag{2.36}$$

that we will denote as A$\mathbf{h} = \mathbf{0}$. A is a 3×9 matrix and $\mathbf{h}$ is a 9-vector composed of components of the unknown matrix H. The component-to-component correspondence between $\mathbf{h}$ and H becomes

$$\mathbf{h} = [\ h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7\ h_8\ h_9\ ]^{\mathrm{T}}, \tag{2.37}$$

$$\mathrm{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}. \tag{2.38}$$

Note that, the matrix A actually has rank two and not three, since for example we can obtain, up to scale, the third row by adding x_i times the first row to y_i times the second row.

Since the homography has eight degrees of freedom up to scale, we need four or more points, whose equations can be stacked on the matrix A to solve for $\mathbf{h}$ using any method that can solve a linear system of the form A$\mathbf{x} = \mathbf{b}$. To obtain a better estimate of the homography, we actually need more than 150 corners [39]. The system is thus over-determined, and because of noise, the matrix A will have a rank larger than eight. We therefore solve in the linear least-squares sense [16, 28].

The linear least-squares solution, however, minimizes the algebraic error $||\mathrm{A}\mathbf{h}||/||\mathbf{h}||$, not the geometric error. What we actually want to minimize is the geometric distance between the points extracted from the image and the points from the scene plane, once transformed using H, or in other words (with properly normalized vectors to avoid trivial solutions)

$$\min_{\mathrm{H}} \sum_{i=1}^{n} ||\mathbf{x}'_i - \mathrm{H}\mathbf{x}_i|| \ , \tag{2.39}$$

which provides the maximum likelihood estimate [16] under the assumption of Gaussian noise. Although this seems to be a normal linear least-squares problem, the scales of $\mathrm{H}\mathbf{x}_i$ are unknown without knowledge of H, so it cannot be solved in the usual manner.

To perform this refinement step known as *bundle adjustment*, we use gradient descent [16, 28], which requires an initial estimate for H, obtained using the DLT algorithm described above. Using such estimated homographies, it is now possible to find the intrinsics of the camera.

2.2.6 Estimation of the Intrinsics

As demonstrated previously by Equation 2.21, to estimate the intrinsics K, we simply need to find the image of the absolute conic $\omega = (\mathrm{K}\mathrm{K}^{\mathrm{T}})^{-1}$, but because the absolute conic is imaginary, it cannot be imaged in reality. However, since we can find homographies between the scene plane and the image plane, we can impose constraints on ω using the circular points via Equation 2.28. The circular points $\mathbf{c}_+$ and $\mathbf{c}_-$ of the scene plane lie on the absolute conic, and therefore their images lie on the image of the absolute conic ω. Using a homography induced by the projection of the scene plane, estimated as described above, we can project the circular points on the image plane and find two equations

$$(\mathrm{H}\mathbf{c}_{+})^{\mathrm{T}}\omega(\mathrm{H}\mathbf{c}_{+}) = 0, \quad (\mathrm{H}\mathbf{c}_{-})^{\mathrm{T}}\omega(\mathrm{H}\mathbf{c}_{-}) = 0, \tag{2.40}$$

which impose two constraints on the image of the absolute conic ω. By Equation 2.2, a conic has five degrees of freedom, and therefore requires five points to be defined. Each image taken from a different configuration (position and angle) provides two such points, so we require a minimum of three configurations and their homographies. In practice, however, we use more than 10 images for best precision [39].

For a given configuration with homography H, if we denote the images of the circular points

$$\mathrm{H}\mathbf{c}_{+} = \begin{bmatrix} x_{c+} \\ y_{c+} \\ w_{c+} \end{bmatrix}, \quad \mathrm{H}\mathbf{c}_{-} = \begin{bmatrix} x_{c-} \\ y_{c-} \\ w_{c-} \end{bmatrix}, \tag{2.41}$$

we can write the two constraints as a linear system in the usual $\mathrm{A}\mathbf{x} = \mathbf{b}$ form, a procedure similar to the one used for the estimation of a homography:

$$\begin{bmatrix} x_{c+}^2 & x_{c+}y_{c+} & y_{c+}^2 & x_{c+}w_{c+} & y_{c+}w_{c+} & w_{c+}^2 \\ x_{c-}^2 & x_{c-}y_{c-} & y_{c-}^2 & x_{c-}w_{c-} & y_{c-}w_{c-} & w_{c-}^2 \end{bmatrix} \vec{\omega} = \mathbf{0} \tag{2.42}$$

that we will denote as $\mathrm{A}\vec{\omega} = \mathbf{0}$. A is a 2×6 matrix and $\vec{\omega}$ is a 6-vector composed of components of the unknown image of the absolute conic ω. Note that each line of the matrix A represents one point on the conic. The component-to-component correspondence between $\vec{\omega}$ and ω becomes

$$\vec{\omega} = [\ a\ b\ c\ d\ e\ f\ ]^{\mathrm{T}}, \tag{2.43}$$

$$\omega = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}. \tag{2.44}$$

We can then use the method described in the previous section, where we solved for a homography, but here we want to solve for $\vec{\omega}$ which has five degrees of freedom up to scale, not eight like a homography. Consequently, we need five or more points, whose equations we write down in the matrix A and solve for $\vec{\omega}$. For best precision, we typically need more than 10 pairs of points [39], and because of noise this renders the system over-determined. As before, we solve in the linear least-squares sense, and the matrix K is finally recovered by applying the Cholesky factorization [16, 28] to ω^{-1}.

Again, the linear least-squares solution minimizes the algebraic error, but we want to minimize the geometric error between points extracted from the image and those from the scene plane, after projection onto the image plane.

Moreover, physical cameras and projectors use lenses that usually exhibit nonlinear *radial and tangential distortions*. We model these using the typical distortion function $d([\ x_u\ y_u\ ]^{\mathrm{T}}) =$

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots) \begin{bmatrix} x_u \\ y_u \end{bmatrix} + \begin{bmatrix} 2p_1 x_u y_u + p_2(r^2 + 2x_u^2) \\ p_2(r^2 + 2x_u^2) + 2p_2 x_u y_u \end{bmatrix} \tag{2.45}$$

where $[\ x_d\ y_d\ ]^T = \mathrm{K}^{-1}\mathbf{x}'$ are the *normalized distorted coordinates* for a point $\mathbf{x}'$ on the image plane, $[\ x_u\ y_u\ ]^T$ are the *normalized undistorted coordinates*, $r^2 = x_u^2 + y_u^2$, $k_{1,2,...}$ are the radial distortion coefficients, and $p_{1,2}$ are the tangential distortion coefficients. As is common practice [5, 51], we only use the first two radial coefficients.

To minimize the geometric error as well as to take into account nonlinear distortion, we perform bundle adjustment using gradient descent [16, 28]. By using the projection model of Equation 2.11 and including the distortion function of Equation 2.45, we minimize (with properly normalized vectors) as follows

$$\min_{\mathrm{K},\mathrm{R},\mathbf{C},k_1,k_2,p_1,p_2} \sum_{j=1}^{m}\sum_{i=1}^{n} ||\ \mathbf{x}'_{ij} - \mathrm{K}\ d(\mathrm{R}_j\ [\ \mathbf{e}_1\ \mathbf{e}_2\ -\mathbf{C}_j\ ]\ \mathbf{x}_i)\ || \tag{2.46}$$

the *reprojection error* for the correspondences between n points $\mathbf{x}_i$ on the scene plane, where $i = 1, ..., n$, and their images $\mathbf{x}'_{ij}$ taken from different configurations $j = 1, ..., m$. We initially assume no nonlinear distortions ($k_1 = k_2 = p_1 = p_2 = 0$). We use as initial estimate for K the matrix found by solving the linear system of Equation 2.42. For R and $\mathbf{C}$, we extract the appropriate elements from the matrix of Equation 2.11. Further, we constrain the skew coefficient s of the matrix K to 0, as this provides better results. Note that for numerical stability, all coordinates $\mathbf{x}_i$ and $\mathbf{x}'_{ij}$ are actually scaled down so their values belong to the range $[-1, 1]$. Resulting matrices are rescaled to fit the original range.

2.2.7 Finding the Intrinsics of a Projector

To find the intrinsic parameters of our cameras, we use the calibration method described above, pioneered by Zhang [51] and refined and implemented by Bouguet [5]. For the projectors, we developed a method based on work done by Raskar and Beardsley [29] and by Ashdown *et al*. [1]. Their methods involve locating points projected onto a calibration board, using the homography between it and its projection onto the image plane of a camera, as described in this section. Ashdown *et al*. [1] use the red and blue channels of a color camera as filters respectively for a calibration board composed of white and cyan squares and for a projected pattern with blue and black squares. This way, one can use the same surface for both a projected and a physical pattern. We opted to use this method, as it maximizes the use of the sensor of a color camera. We tested cyan, magenta, yellow, red, green, and blue colors under complementary projector light, and confirmed that cyan provided the best precision. Since we do not employ mobile projectors as Ashdown *et al*. [1], we calibrate a projector by holding our cyan calibration board, a printed sheet attached to a foam board, at different angles in front of the projector as depicted in Figure 2–4. We found that our projectors were too powerful and emitted too much blue light in comparison to the ambient red light of the room. However, rather than simply reducing the intensity of the projector to fall within the dynamic range of the camera sensor, we took advantage of the fact that projector light is typically more powerful than room lighting. Therefore, we use the projector to light up the calibration board with red and green light as well, resulting in a projected pattern with yellow and white squares. This way, we are able to guarantee the ratio of blue pixel intensity to red pixel intensity without relying on the environment or fine-tuning camera parameters. Typical images are shown in Figure 2–4.

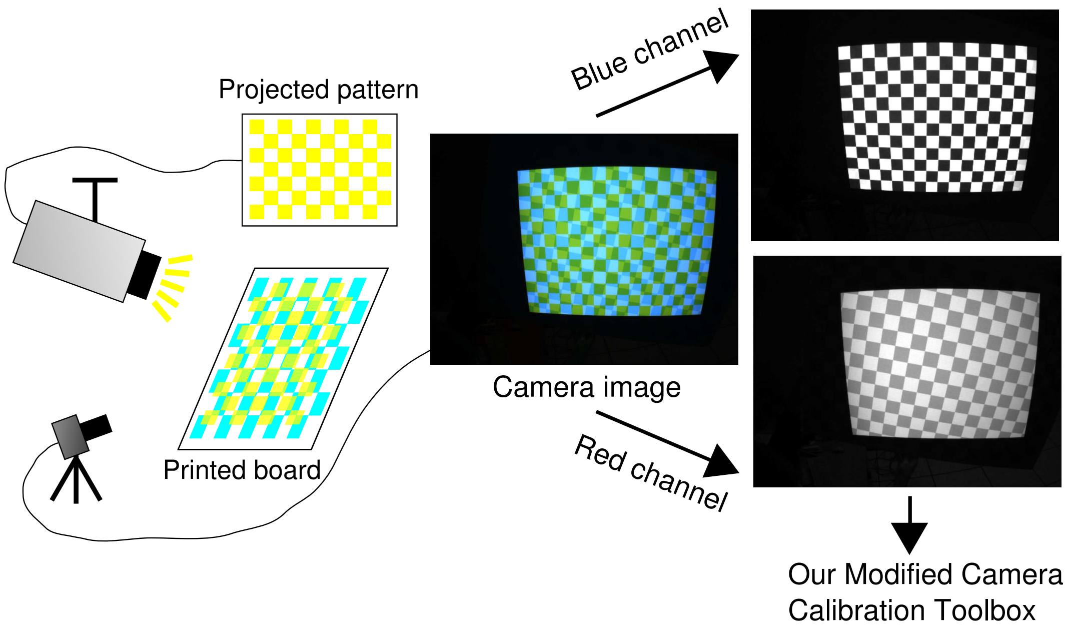


Figure 2–4: For projector calibration, we hold the cyan calibration board in front of the paired camera and the projector displaying the yellow pattern as shown at left. Sample images produced by this procedure are shown at right. (The level of the red channel is adjusted for better contrast.)

Next, the yellow projected pattern predetermines the m points $\mathbf{x}'^{\mathrm{p}}_k$ on the image plane of the projector, where $k = 1, ..., m$. Via the red and blue camera images, we can find their corresponding physical location $\mathbf{x}^{\mathrm{p}}_k$ on the calibration board, which is the data required for calibration. To do so, we first use the corner extraction method, detailed in Section 2.2.4, on the red and blue channels to find the sets of points $\mathbf{x}^{\mathrm{red}}_i$ and $\mathbf{x}^{\mathrm{blue}}_k$. The $\mathbf{x}^{\mathrm{red}}_i$ are actually the camera images of the n corners on the calibration board $\mathbf{x}_i$ where $i = 1, ..., n$, which we previously denoted simply as $\mathbf{x}'_i \equiv \mathbf{x}^{\mathrm{red}}_i$, while the corners from the blue channel are the camera images of the m points $\mathbf{x}^{\mathrm{p}}_k$ we are looking for.

Therefore, using the procedure described in Section 2.2.5 on $\mathbf{x}_i^{\mathrm{red}}$ and $\mathbf{x}_i$ we can find the homography H_{bc} between the board and the image plane of the camera, such that the following hold:

$$\mathbf{x}_i = \lambda_i \mathrm{H}_{\mathrm{bc}} \mathbf{x}_i^{\mathrm{red}} \ , \tag{2.47}$$

$$\mathbf{x}_k^{\mathrm{p}} = \lambda_k^{\mathrm{p}} \mathrm{H}_{\mathrm{bc}} \mathbf{x}_k^{\mathrm{blue}} \ , \tag{2.48}$$

for arbitrary scaling factors λ_i and λ_k^{p}.

We also have to take into account the known nonlinear distortion coefficients of the camera and use the undistorted coordinates as described by the function of Equation 2.45.

Finally, to calibrate a projector, we use the predetermined point images $\mathbf{x'}_k^{\mathrm{p}}$ and the points $\mathbf{x}_k^{\mathrm{p}}$ found using the steps described in this section. The rest of the calibration procedure is identical to the one used for the cameras, described in the previous section, substituting $\mathbf{x}_i$ by $\mathbf{x}_k^{\mathrm{p}}$, and $\mathbf{x'}_i$ by $\mathbf{x'}_k^{\mathrm{p}}$.

We also pay attention to other important factors that can influence the accuracy of the resulting calibration. Previous research [39] indicates that to obtain near optimal results with the method of Zhang [51], the calibration board should have more than 150 corners and the set of images should contain more than 1500 corners, and for these reasons we do the following. We use at least 15 images for the calibration of each device. The calibration board used for the cameras contains 18 × 14 squares of 15 mm (221 corners), the cyan calibration board includes 19 × 14 squares of 50 mm (234 corners), and the projected pattern is made of 16 × 12 squares (165 corners). We hold the calibration boards at angles near 45° as recommended by Zhang [51]. Last, when calibrating a projector, we install the camera on top of it so that the projected pattern image can use the largest possible area of the camera sensor even when holding the board at different angles.

2.2.8 Extrinsic Parameters

After obtaining the intrinsic parameters of the cameras and projectors, we need to find the extrinsic parameters. As described earlier in Section 2.1.2, the scene plane induces a homography between it and the image plane of a *projective device* (camera or projector). Therefore, the scene plane also induces a homography between the image planes of two projective devices. In the present case, we use the flat display surface as the scene plane, as shown in Figure 2–5. For each projector, we find the homography between it and the camera by first projecting a known calibration pattern on the display surface. Again, we have n predetermined points $\mathbf{x}'^{\mathrm{p}}_i$, where $i = 1, ..., n$ in the image plane of the projector. Then, we locate the corresponding points $\mathbf{x}'_i$ in the image plane of the camera using corner extraction, and we determine the homography by using the algorithm described in Section 2.2.5. For two calibrated devices, an algorithm developed by Triggs [44] can then decompose the homography into the rotation matrix and translation vector separating the two devices, and also provides the equation of the inducing plane. The algorithm actually returns two sets of values, only one of which is true. In our current setup, the display surface remains the same for at least two projectors, so the actual plane equation has to be the same for these

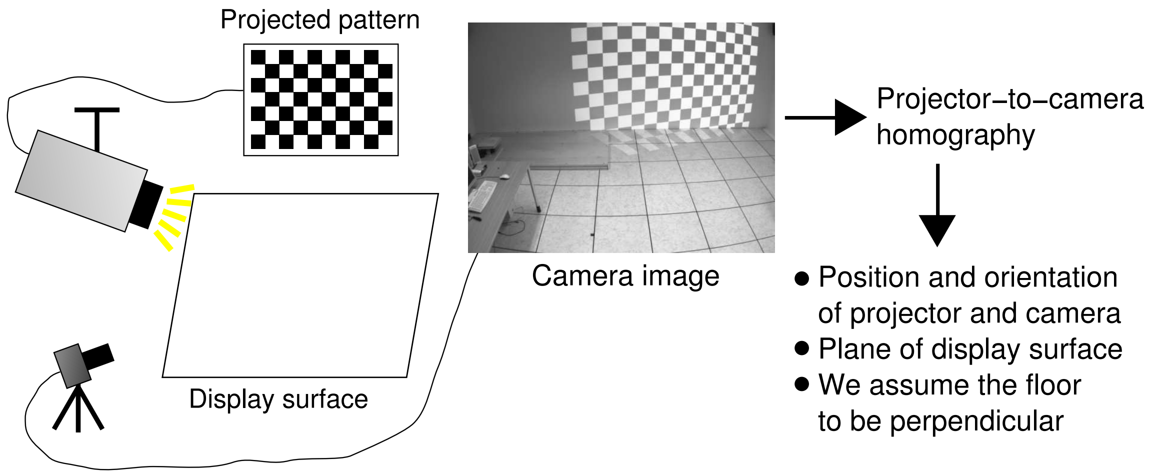

Projected pattern
Display surface
Camera image
Projector–to–camera homography
• Position and orientation of projector and camera
• Plane of display surface
• We assume the floor to be perpendicular

Figure 2–5: Overview of the method used to find the extrinsics of the cameras and projectors, and the planes of the display surface and the floor. The projector projects a checkerboard pattern on the wall, and with the corners extracted from the camera view, we can find a homography induced by the display surface, which we decompose to find the required information.

projectors. We can therefore easily resolve the ambiguity and retain the set of values for which the plane equations are closest to one another.

Because of unavoidable errors in the intrinsic parameters and in corner extraction, the plane equations are not exactly equal. Without a unique plane, we cannot properly model the shadows and back-project them onto the display surface from two or more devices. Furthermore, we do not want to lose the accurate homographies between projectors (less than one camera pixel of error), since they indicate how well the content displayed by the projectors can overlap. To satisfy both goals, we devised the following procedure. First, we use the plane equation $\mathbf{p}_c$ found with the homography between two cameras. We chose to use this plane since the calibration procedure provides more accurate intrinsic parameters for cameras than for projectors. Thus a plane found with cameras should be more accurate than one found with a camera and a projector. Next, the idea is to find a corrective homography H to remove the discrepancies between the predetermined images $\mathbf{x}'^{\mathrm{p}}_i$ and the projection of corresponding 3D points $\mathbf{X}_i$ from the display surface $\mathbf{p}_{\mathrm{c}}$ onto the image plane of the projector, which should be equal, or in other words:

$$\mathbf{x}'^{\mathrm{p}}_i = \lambda_i \mathrm{H}\mathrm{P}_{\mathrm{p}}\mathbf{X}_i \tag{2.49}$$

where

$$\mathbf{X}_i = \mathrm{B}_{\mathrm{c}}\mathbf{x}'_i \tag{2.50}$$

where P_{p} is the projection matrix of the projector, $\mathbf{x}'_i$ are the points extracted from the camera image, and B_{c} is the back-projection matrix of the camera, as described in Section 2.1.3, using as plane $\mathbf{p}_{\mathrm{c}}$.

From this operation, H is the corrective homography we want to find to obtain a corrected matrix $\mathrm{P}'_{\mathrm{p}} = \mathrm{H}\mathrm{P}_{\mathrm{p}}$, which can project points $\mathbf{X}_i$ to $\mathbf{x}'^{\mathrm{p}}_i$ as closely as possible in the linear least-squares sense. We ignore the radial and tangential distortion coefficients of the projector found during calibration. This approximation should be valid as the projectors do not exhibit noticeable distortion. Also, this simple procedure as a whole does not attempt to minimize geometric errors induced by changes to the projection matrix. However, assuming relatively accurate calibration results, the correction by the homography H, and consequently the errors, should be minimal.

At this point, although we have all the parameters of the cameras, projectors and display surface, the tracking module requires the equation of the floor plane and any other planes needed to model the room. For our current setup, since no other planes enter the field of view of the cameras, we only need the floor plan. To obtain it, we let the user manually determine in a physical unit the edge vector $\mathbf{v}$ in the camera image where the floor and the display surface meet. Assuming they are at right angles, we find the normal of the floor $\mathbf{n}_\mathrm{f}$ by computing the cross-product of the normal of the display surface $\mathbf{n}_\mathrm{d}$ and the edge vector $\mathbf{v}$, or in other words $\mathbf{n}_\mathrm{f} = \mathbf{n}_\mathrm{d} \times \mathbf{v}$. We also rescale all calibration parameters with the physical length of the vector $\mathbf{v}$. Once we have all this information, we can use it to track the positions of people in the room, as described in the next chapter.

CHAPTER 3
Tracking

Before we can perform shadow removal, our model requires 3D tracking information about people in the scene. There are many tracking algorithm that can be used to obtain this information, but we were motivated to use an approach we could easily implement and that would be fast enough for interactive purposes. We also wanted to test our system with a method that did not require the user to wear sensors, tags or markers, and that could also work in an environment filled with video projections. These are goals that can be attained using computer vision, even though current methods are not entirely satisfactory. Rather than focusing our efforts on developing a new tracking method or on implementing a complex 3D tracking algorithm [49], we devised a simple method based on previous research using computer vision [20, 38], which is sufficient for evaluation purposes of the shadow removal process. The method involves computing a background disparity map between two calibrated color cameras and then using the map on images from the cameras to obtain disparity contours. An object tracker can then use these images directly for tracking people in the scene.

3.1 Disparity Contours

Using images from two cameras we can generate disparity contours. These have the property of being less sensitive to changes happening on the display surface than normal unprocessed camera images. To compute disparity contours, we first need the calibration data of the cameras and the room (floor and display surface) to build a *background disparity map*. This map can be seen as a function $\mathbf{x}_i^2 = m(\mathbf{x}_i^1)$ that maps points $\mathbf{x}_i^1$ from the image plane of one camera into points $\mathbf{x}_i^2$ on the image plane of the second camera, where $i = 1, ..., n$. As described by Ivanov *et al.* [20], with a pair of frames I^1 and I^2, and their disparity map, one can compute a difference image

$$\mathrm{D}(\mathbf{x}_i^1) = |\mathrm{I}^1(\mathbf{x}_i^1) - \mathrm{I}^2(m(\mathbf{x}_i^1))| \tag{3.1}$$

for $i = 1, ..., n$. In theory, $\mathbf{x}_i^1$ and $m(\mathbf{x}_i^1)$ should back-project to the same physical point when no object is present in front of the cameras, thus producing a completely black difference image D regardless of changes in lighting conditions under the assumption of Lambertian surfaces. However, when the disparity map does not correspond to the current scene geometry, this procedure produces contour images, referred to as *disparity contours* [38], which have an appearance similar to the output of an edge detector. Although in theory it is possible to extract depth information from those contours, the task has proven challenging [38], and we instead use the information for 2D tracking only.

Before routing these images to a blob tracker [7], we erode the images with a 3×3 square mask, and then resize them to half their width and height for two reasons. First, since we are dealing with discrete pixels, there are always registration errors of at least ± 0.5 pixels even assuming a perfect disparity map. Second, the tracker requires much processing power and its limited precision did not warrant that we use full resolution.

We then send the eroded and resized images to the blob tracker. In practice, these images are not completely black when no object is present. There are always some calibration errors as well as differences in intensity because of non-Lambertian surfaces and inconsistencies between CCDs. As a consequence, we use a second preprocessing stage where we chose the Mixture of Gaussian background subtraction algorithm [24], which works best at minimizing the importance of the various causes of errors. The remaining steps of the tracking algorithm remain unmodified. The results are passed to a Kalman filter [48] that models the speed of the targets to predict their future locations. We decided to have the filter run two prediction steps, accounting for the delays in the disparity contour and tracker modules as well as in the shadow removal and image rectification modules, assuming the two delays are close to each other. In this manner, we obtained an object tracker less sensitive to changes on the display surface, as we can see in Figure 3–1 where the image on the wall hardly shows up in the disparty contours.

3.2 3D Tracking

Although the low complexity of this tracking method is a desirable feature, it provides only 2D information and does not manage occlusion well. Solutions to the latter problem are beyond the scope of this work, but a possible workaround could be to install the cameras on the ceiling. However, with a large field of view and multiple people, occlusions are more likely to be observed, which would create problems for the tracker. To address the limitations of the 2D tracker, we estimate 3D information as follows.

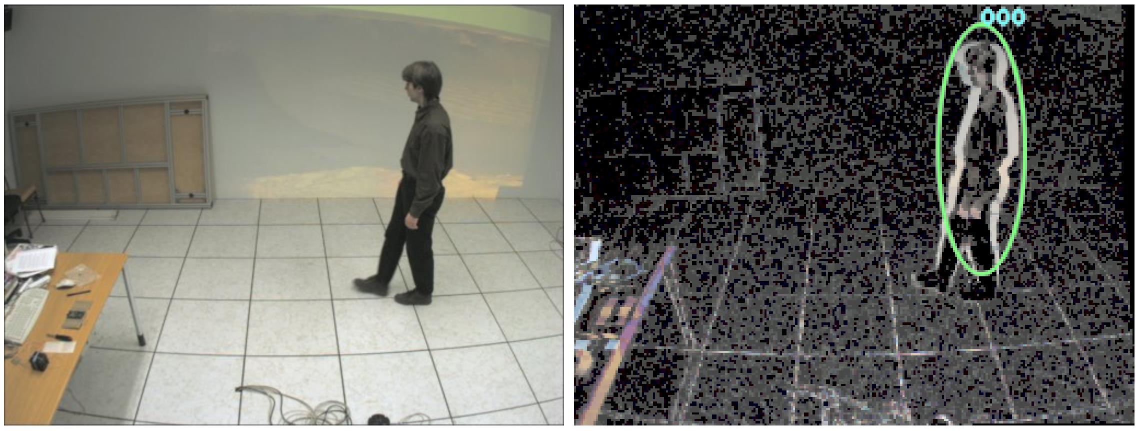

(a) Camera image. (b) Disparity contours and tracking.

Figure 3–1: Sample camera image and the corresponding disparity contour image with tracking information. (The level of the disparity contours is adjusted to easily see details of low brightness.)

As described in Section 2.1.3, we can back-project a point $\mathbf{x}'$ corresponding to the feet of a person in the image plane of the camera onto a point $\mathbf{X}$ on the floor plane $\mathbf{p}_{\text{floor}} = [\ a\ b\ c\ d\ ]$. We can then approximate the height of people, assuming they stand in the direction of the normal of the floor (*i.e.* vertically). Consider a plane $\mathbf{p}_{\text{top}} = [\ a\ b\ c\ e\ ]$ parallel to the floor right above a person. It consequently has the same normal $\mathbf{n} = [\ a\ b\ c\ ]^{\mathrm{T}}$ as the floor plane $\mathbf{p}_{\text{floor}}$. Therefore, the distance we are interested in is the height $h = d - e$, after normalization. The back-projection matrix B in Equation 2.18 has rank three, so it can solve for three unknowns, but we only have two unknowns: the height h and the scale of the resulting $\mathbf{X}$, so the system is overdetermined. We could have found the least-squares solution to minimize errors with $\mathbf{x}' = [\ u\ v\ 1\ ]^{\mathrm{T}}$, but under the assumption that the up vector of the camera is in the general direction of the normal, simply dropping the u component and using only the v component should produce acceptable results. To find the width of the person, we simply back-project to the floor the two bottom points of the bounding box of the 2D tracking information, and calculate their difference.

Similarly in 3D, we decided to model people as flat rectangles parallel to the display surface, using as height and width the values found by the procedure described in the previous paragraph. For our current setup, these simplifying assumptions yield acceptable results, which we use as input to the shadow removal module as described in the next chapter.

CHAPTER 4
Shadow Removal

The shadow removal module uses the calibration and tracking information as found using the methods described in Chapters 2 and 3. Figure 4–1 shows a rendered image of the model of our room, as described by calibration and tracking data. This model contains two cameras, two projectors, a tracked person, the floor, the display surface, as well as modeled shadows rendered over it in the color of the occluded projectors. To compute these shadows, the system simply back-projects the tracked people that are represented by rectangles onto the display surface, as described previously with Equation 2.13.

For a given projector, we can understand these shadowed regions as a *coverage mask*, which is an image where white pixels indicate the absence of a shadow according to the model, and black pixels, the presence of a shadow. The system uses these coverage masks, one for each projector, as the basis for the rest of the shadow removal process. The following processing stages include intensity blending, normalization, and image rectification. Intensity blending smooths transitions between projectors since their properties differ from one to the next. Then, we perform normalization, which assumes uniform projection intensity within a single projector. Even though this is not actually the case with typical projectors, this produces acceptable results. Under this assumption, normalization ensures that the intensity for any given region on the display surface remains

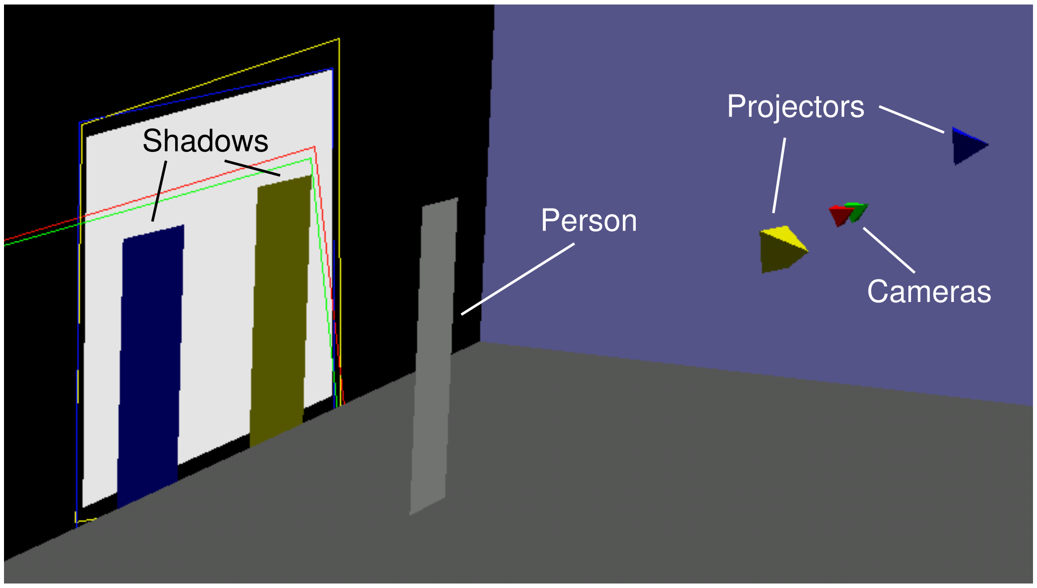
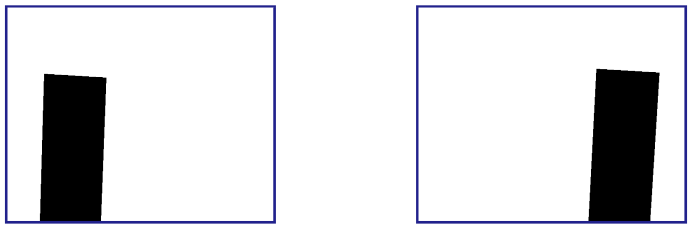


Figure 4–1: Rendered image of the 3D model of the room with cameras, projectors and a tracked person modeled as a rectangle, where the floor is in dark gray, the wall, in black, and its chosen display surface in white. The lines on the wall denote regions that cameras can see and that projectors can cover, drawn in their corresponding colors.

constant, although color calibration would be necessary to obtain optimum results. Finally, to account for projector distortion (linear keystone, and non-linear radial and tangential distortions) image rectification pre-warps the display content and the masks to obtain the desired rectified result on the display surface. We implemented this operation in OpenGL to execute it quickly on graphics hardware.

4.1 Projector Coverage

Shadow removal works by first creating a coverage mask image for each of the n projectors. A mask defines a portion of the display surface where images are to be displayed, with the shadow regions removed. For the sake of clarity, we use $[0,\ 1]$ as the range of pixel values for discussion in this section, although our



(a) Projector 1. (b) Projector 2.

Figure 4–2: Sample coverage images generated for the model of Figure 4–1.

implementation scales the results appropriately for use with 8-bit grayscale images. A pixel with a value of 0 means that it is off, and 1, that it needs to be fully lit by the projector corresponding to the mask.

First, the algorithm sets all the pixels of the mask images to 0. Then using the calibration information, it sets to 1 the pixels inside the quadrilateral region that is coverable by the projector. At this point, the images contain only information about which area of the display surface any particular projector can cover. Then, we need to localize shadows using tracking and calibration information. We can model shadows similarly to how the operation is done in computer graphics. For each projector, the algorithm back-projects the vertices of the rectangle onto the display surface as described previously in Section 2.1.3, and fills in the affected regions of the mask images with zero-valued pixels, generating masks such as those shown in Figure 4–2.

Next, we want to normalize each pixel so that the sum of all pixels in the same position from all masks M_j does not exceed 1, as described by the following

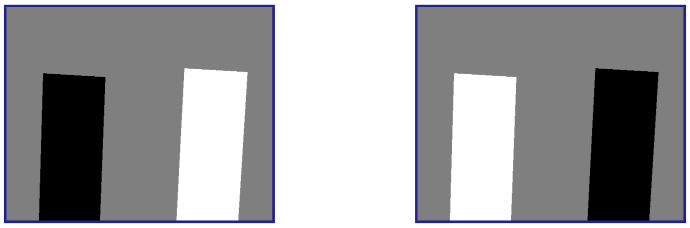

(a) Projector 1. (b) Projector 2.

Figure 4–3: Sample result of normalization for coverage images of Figure 4–2.

equation:

$$\hat{\mathrm{M}}_j(u,v) = \begin{cases} 0 & \text{if } \sum_{k=1}^{n} \mathrm{M}_k(u,v) = 0 \\ \frac{\mathrm{M}_j(u,v)}{\sum_{k=1}^{n} \mathrm{M}_k(u,v)} & \text{otherwise} \end{cases} \quad (4.1)$$

for $j = 1, ..., n$ where M_j is the mask image before normalization, and $\hat{\mathrm{M}}_j$ after, for all pixels (u, v). We show a sample result of normalization in Figure 4–3.

Note that if an occluder stands too close to the display surface, this algorithm cannot remove the shadow completely, and pixels in problematic portions would be given a value of 0 in all masks. In this case, instead of setting all such pixels to 0, the algorithm actually finds the first projector that can cover this area in a non-occluded situation, and sets the pixels to 1 in the corresponding mask. This way, any small portion that is not actually occluded is still displayed. Also, projectors contribute equally to as much of the display as possible. This maximizes the likelihood that in the event of a failure of shadow removal, the displayed content will at least remain visible, even if reduced in intensity.

4.2 Intensity Blending

Although normalization alone would be sufficient with ideal projectors, in practice it is not. Manufacturing limitations result in projectors often having different color emission properties. Projection angle also affects the quality and consistency of multi-projector displays. As seen in Figure 4–3, the masks have sharp edges at the coverage limit, and simply projecting them at this point would make these edges stand out to the human eye. To reduce this distracting effect, we added a smoothing stage that softens the transition between different projectors. We perform *intensity blending* as first described by Raskar *et al.* [32] to smooth out these edges, without changing the overall intensity of the projection.

We first apply a *distance transform* on the coverage image of each projector. An approximation of the true distance transform can be computed efficiently in $O(n)$ time [4], where n is the number of pixels. The distance image produced for one projector indicates how far away each pixel is from an edge of its coverable region. We then normalize these results with a chosen maximum pixel value $d_{\max}$, equal to, for example, 50, which indicates within how many pixels we want blending between multiple projectors to take place. Denoting M_j^d the distance transform of coverage image M_j, this can be formulated mathematically for all pixels (u, v) as

$$\bar{\mathrm{M}}_j^d(u,v) = \begin{cases} d_{\max} & \text{if } M_j^d(u,v) > d_{\max} \\ M_j^d(u,v) & \text{otherwise} \end{cases} \tag{4.2}$$

$$\hat{\mathrm{M}}_j^d(u,v) = \begin{cases} 0 & \text{if } \sum_{k=1}^{n} \bar{\mathrm{M}}_k^d(u,v) = 0 \\ \frac{\bar{\mathrm{M}}_j^d(u,v)}{\sum_{k=1}^{n} \bar{\mathrm{M}}_k^d(u,v)} & \text{otherwise} \end{cases} \tag{4.3}$$

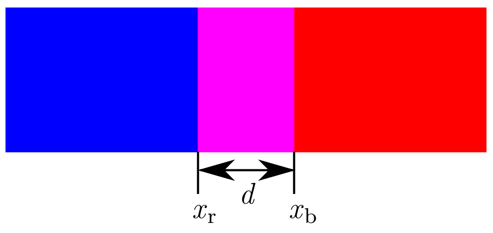


Figure 4–4: The blue region on the left and the red region on the right represents two projections, with their overlapping region of length d. The border of the "blue" projector is at x_{b} and the one for the "red" projector is at x_{r}.

where $\hat{\mathrm{M}}_j^d$ is the final blended and normalized mask image for projector j. Normalization ensures that the overall intensity remains constant over the whole display surface.

To see that this operation actually smooths the transition regions, first consider the limits of Figure 4–4. A pixel just outside the display region of one projector, such as the "blue" projector at x_{b}, receives zero illumination from it, but if another (*e.g.*, "red") projector covers this pixel, it may receive full illumination. The same reasoning applies at the other end, such as at x_{r}, at the display limit of the "red" projector. Next, the value of the distance image decreases by one for each pixel closer to the limit, such that the relation between the pixel value and the distance is linear. Thus, applying the distance transform to these overlapping regions results in a linear smoothing effect. Similar reasoning using unit hyperspheres holds for higher dimensions.

In practice, this method produces visually pleasing smooth transitions for an arbitrary number of projectors [32], shadows not affecting in any way the definition of the problem. Figure 4–5 shows a sample result of blending.

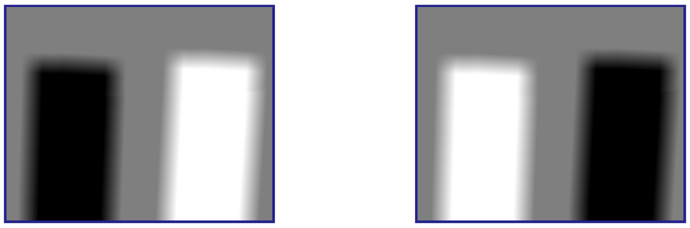

(a) Projector 1. (b) Projector 2.

Figure 4–5: Sample result of intensity blending via distance transform and normalization for coverage images of Figure 4–2.

4.3 Image and Mask Projection

The masks can then be used to modulate the brightness of the projector pixels. However, this linear representation of intensity is not physically reproduced by commodity projectors. They usually follow a power law of 2.2 [19]. To compensate, we filter the masks using a function that performs a gamma correction of $\frac{1}{2.2} = 0.45$, *i.e.*: $\hat{\mathrm{M}}'(u,v) = \hat{\mathrm{M}}(u,v)^{0.45}$ for all pixels (u,v). Next, we need to rectify the images so that they appear geometrically correct on the display surface. Note that from this point on, the processing required for a projector is independent of all others, such that the following operations can be executed in parallel on multiple computers.

4.3.1 Image Rectification

At this point, we have a set of masks and an image as display content, but they represent what should appear on the display surface, not what should be sent to the projectors. If we consider Figure 4–6(a) as an image or mask to display, we first normalize the image dimensions such that they correspond to the ones of the display surface, Figure 4–6(c). Then, we essentially want a method that allows us to keep this identity relationship. We know from calibration the homography

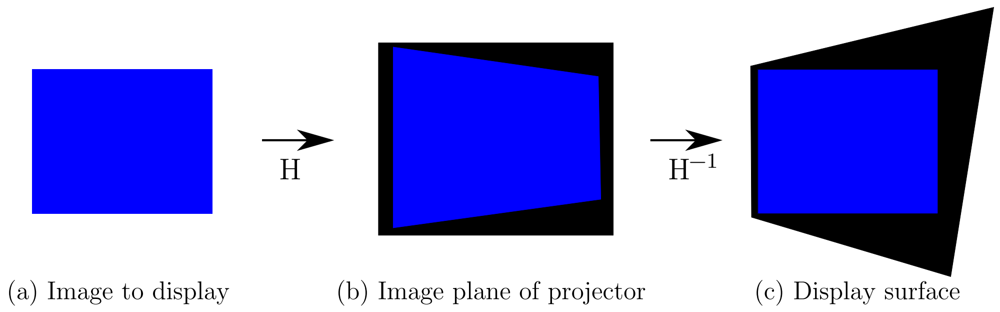


Figure 4–6: The transformations an image or a mask has to go through before appearing on the display surface. Note that the blue image at (c) has the same dimensions and shape as (a).

H^{-1} that the projector induces by placing pixels onto the display surface, which is the inverse of a projective transform H that a dual camera would use to image points from the surface. Consequently, we would like to find and apply the inverse $(H^{-1})^{-1} = H$ before sending an image to the projector such that $H^{-1}H = I$. As seen previously in Section 2.1.3, we can easily find the projector corners on the plane of the display surface using back-projection. We also know the dimensions of the image plane of the projector, and thus its four corners. This produces a four point correspondence which we use to find the homography H using to the DLT algorithm of Section 2.2.5. One can then use this homography to pre-warp images. For efficiency, we implemented image transformation using OpenGL as described next.

4.3.2 Image Transformation with OpenGL

The usual way to apply an arbitrary transformation to an image is by first computing an *inverse mapping function* that takes coordinates $\mathbf{x}'$ and transforms them back into the corresponding coordinates $\mathbf{x}$ in the source image:

$$\mathbf{x} = m(\mathbf{x}'). \tag{4.4}$$

Programmatically, this function is optimally implemented using a *look-up table*. With this function or table, to fill the transformed image, we simply have to loop over all of its pixels, and use the mapping function to locate the appropriate values from the source image. The function could also return a weighted combination of source pixels to consider, which makes it possible to implement optimized versions of other interpolation methods.

However, since the mapping works on a pixel-by-pixel basis, it can be slow. There exists a more efficient method using OpenGL and a Graphics Processing Unit (GPU) [6, 41, 43]. OpenGL and GPUs were built to handle large amounts of graphics data effectively. We can feed an image, called a *texture*, to the graphics hardware through OpenGL and instruct it to perform transformations.

Instead of pixels, the basic processing primitive of OpenGL and of graphics hardware is triangles. A transformation is therefore applied to a whole triangle at a time, not a single pixel, which dramatically increases performance. However, the range of possible geometric transformations is limited to linear ones (homographies). We can use two triangles to warp a rectangle, the shape we are most interested in, but as shown in Figure 4–7, the resulting transformation does not equal the desired one. In Figure 4–7(c), we can clearly see the boundaries of the two triangles where the bottom line and the horizon line at the top are not straight.

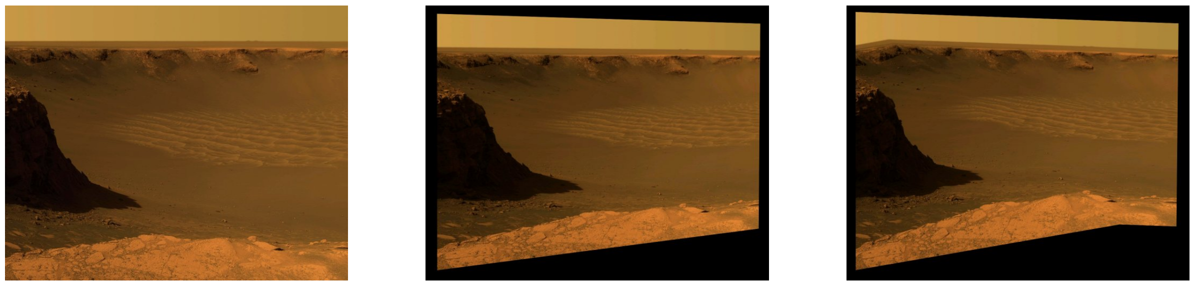
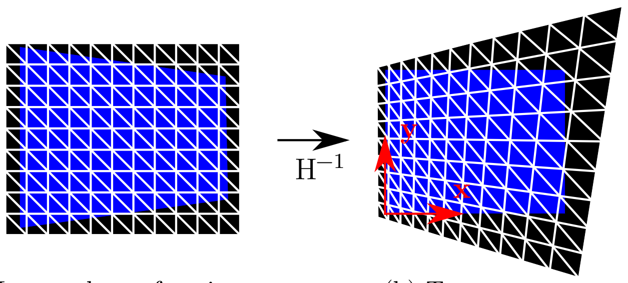

(a) Sample image. (b) Desired transformation. (c) With two triangles.

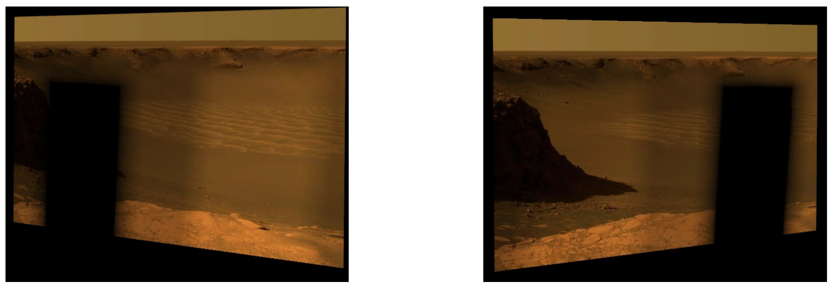
Figure 4–7: Sample image and its desired transformation using OpenGL. Only two triangles do not provide satisfactory results.

To overcome this limitation, we can divide the image plane of the projector into a triangular mesh and instruct OpenGL to transform each small triangle piece by piece from the texture. As shown in Figure 4–8, we apply the inverse transformation H^{-1} to the vertices of the triangles in the image plane. In other words, to each vertex $\mathbf{v}$, we assign the coordinates $\mathrm{H}^{-1}\mathbf{v}$ in texture space. Then, when we send a new texture, OpenGL transforms it into the image plane of the projector, and we obtain the desired transformation H. This is analogous to the inverse mapping function described earlier. This way, we efficiently achieve any arbitrary transformation, including linear (projective) as well as nonlinear (radial and tangential) distortions. The errors of incorrectly transformed pixels within each triangle reduce as the mesh is refined, but processing delay also increases as each triangle converges to a single pixel. Tardif *et al.* [43] used a grid of 12065 squares, and in practice we also found a mesh of 101×101 vertices to be visually sufficient for typical images of 1280×1024 pixels. Figure 4–9 shows a sample rectification of the display content with intensity blended masks layered on top.



(a) Image plane of projector. (b) Texture space.

Figure 4–8: The image is divided into a triangular mesh, and the inverse transformation H^{-1} is applied on the vertices, into the texture space.



(a) Projector 1. (b) Projector 2.

Figure 4–9: Sample result of image rectification for the blended masks of Figure 4–5 layered on top of the sample image of Figure 4–7. These are typical images sent to the projectors.

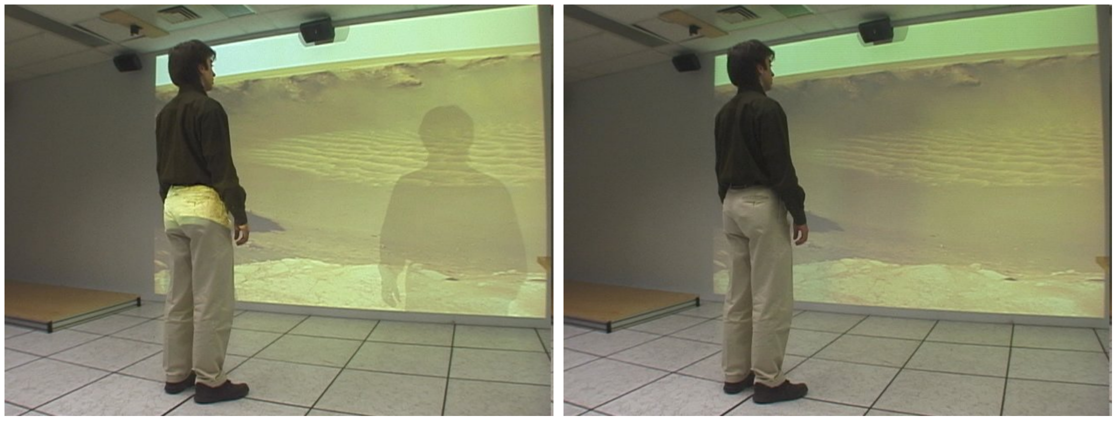
With this, the description of our shadow removal method is complete, and we present results in the next chapter.

CHAPTER 5
Results

We tested our system in the Shared Reality Laboratory where a front projection system is installed to project on the interior wall surface of the room. We used two Sanyo PLC-EF30 projectors and two Point Grey Research Flea2 cameras equipped with 4.0-12.0 mm varifocal lenses adjusted to their widest angle. These cameras automatically synchronize with each other.

In Figure 5–1, we show qualitative sample results with and without shadow removal, a concrete illustration of the benefits of our system. Without, we can see the dimmer area on the display surface as well as projector illumination on the person as an occluder. With shadow removal, the dimmer region as well as the extra illumination on the person disappears. Note that because of color differences between the two projectors, the left side of the displayed content exhibits a blue bias while the right side is more red. Although this artefact should ideally be corrected, it demonstrates that our shadow removal algorithm is working correctly, blending the output of the two projectors as intended. Nevertheless, for improved blending, we would need to perform color calibration on the projectors [3, 17, 18, 21, 22, 23, 37].

In the rest of the chapter, we explain the tests we ran and the numerical results obtained. The experiments involved three distinct steps: calibration, tracking, and shadow removal, each of which is described in detail below.



(a) Without shadow removal. (b) With shadow removal.

Figure 5–1: Sample images of the room with one person standing, with and without shadow removal.

5.1 Calibration

Using our modified version of Bouguet's calibration software [5], the reprojection errors (root mean square errors in pixels) achieved for the intrinsics after calibration of the cameras and projectors are shown in Table 5–1. We performed the reprojection using the native resolution of each device as defined during calibration. The resolution of the cameras was 1024×768 and that of the projectors was 1280×1024. During projector calibration, for consistency, we used images and calibration results from Camera 1 only. As for the extrinsic parameters, Table 5–3 describes the position and orientation in space as found by our method, using reference vectors defined in Table 5–2.

After incorporating corrective homographies, the disparity between corresponding points from cameras or projectors all dropped to less than one pixel, as expected, according to the initial homographies. Interestingly, although this correction altered the rotations and translations of the projection matrices, it

Table 5–1: Reprojection error (root mean square error in pixels) after calibration.

	Reprojection error	
	u	v
Camera 1	0.274	0.250
Camera 2	0.251	0.212
Projector 1	0.724	0.777
Projector 2	0.499	0.543

Table 5–2: Reference vectors for the floor, the wall, and the projective devices (cameras and projectors), all axes following the right-hand rule convention.

	Reference Frame	Vector
Floor normal	World	$[\,0\ \ 0\ \ 1\,]^{\mathrm{T}}$
Wall normal	World	$[\,0\ -1\ \ 0\,]^{\mathrm{T}}$
Up vector	Device	$[\,0\ \ 0\ \ 1\,]^{\mathrm{T}}$
Principal axis	Device	$[\,0\ \ 1\ \ 0\,]^{\mathrm{T}}$

Table 5–3: External parameters of the projective devices in the room.

	Position (cm)			Orientation (degrees)		
	x	y	z	Roll	Pitch	Yaw
Camera 1	−22.5	−425	215	−2.65	−28.3	−8.00
Camera 2	−6.28	−429	215	−2.18	−29.2	−5.61
Projector 1	226	−451	227	−1.74	−5.54	11.7
Projector 2	−94.4	−418	202	−2.97	−4.68	−24.4

affected the internal parameters to a much higher degree. Even though the corrective homographies are close to the identity matrix, the procedure described in Section 2.2.8 minimizes algebraic error, rather than the geometric error. Without proper normalization, such results are not surprising.

Next, we verified the accuracy of the projector calibration by having a person stand at the corners of every tile (61×61 cm) in the room, positions indicated by the x row and y column of Tables 5–4 and 5–5. The real height and width of the test subject were 175 cm and 53 cm respectively. Although we used the exact position on the floor, to account for the non-flatness of people we padded the height and width measurements to 180 cm and 60 cm, respectively, as seemed appropriate based on the thickness of the person. Afterwards, we asked the subject to stand at the exact y positions as indicated by the predetermined locations on the floor, but with the x positions chosen to obtain the smallest area of shadow. During this exercise, we made physical measurements of the errors. It was always possible to find a position where the width of the shadow appeared to be correctly modeled by the padded dimensions. It seems that there were systematic errors in all axes, which increased as the person stood further away from the origin, suggesting biased calibration errors. Table 5–4 summarizes the differences in x between exact desired location and physical measurements. A portion of the subject's head sometimes cast a shadow, so we measured that portion, yielding additional information on calibration accuracy. This difference in z is shown in Table 5–5. Although we accumulated data for both projectors, each cell only has one value, since the shadow was always fully corrected for at least one of the projectors. The maximum error we found on the floor was 54.3 cm, and the maximum error in z, 12.1 cm, while on average the errors were 28.0 cm and 2.0 cm respectively. We repeated each measurement several times to confirm that all measurements were precise within 2.0 centimeters.

Table 5–4: Difference in x (cm) between exact desired location and physical measurements ($\pm$1.0 cm).

		x (cm)				
		0	61	122	183	244
y (cm)	−61	15.1	14.9	14.9	15.3	16.2
	−122	15.3	16.4	18.9	21.3	23.7
	−183	20.2	26.5	29.6	29.7	32.4
	−244	30.5	32.7	34.3	35.2	36.8
	−305	36.9	43.1	43.2	43.6	54.3

Table 5–5: Difference in z (cm) between exact desired height and physical measurements ($\pm$1.0 cm). Empty cells indicate that the shadow cast by the head on the wall was fully corrected. "N/A" means that there was no occlusion in front of either projector.

		x (cm)				
		0	61	122	183	244
y (cm)	−61					
	−122					1.7
	−183			1.7	2.5	2.7
	−244	0.1	2.0	5.9	6.4	6.3
	−305	4.6	N/A	N/A	11.5	12.1

More recent experiments with a DeepSea V2 Stereo Camera from Tyzx [49] having a baseline of 22 cm and a field of view of 75° suggest however that we were able to obtain an accuracy within the margin of error of the camera, which is approximately 5 cm at 3 m. In this setup, we use only one camera for calibrating the extrinsics. It would appear that the assumption we made earlier in Section 2.2.8 is wrong. We thought that a plane found with two cameras should be more accurate than one found with a camera and a projector, since the calibration of the cameras is more accurate. However, the distance separating two devices obviously has a great influence on the accuracy of the extrinsic parameters.

The consequence of the errors in calibration described in this section is that to compensate we have to use a larger padding than the arbitrary one we chose before executing the tests. It also means that the initial choice of padding did not affect this conclusion. Moreover, despite the errors, our implementation performed well as described later in Section 5.3. The tracking module is another source of errors, as explained next.

5.2 Tracking

The focus of our work was not to compare different tracking algorithms, and we simply adopted the blob tracker from the OpenCV Video Surveillance Module [7]. Even though the disparity contours help make the tracker insensitive to changes on the display surface, compared to the area of the person in the original image, the area of the resulting contours is small. In our experiments, the disparity map was accurate within one pixel, but color response differences between the two cameras meant that the blob tracking algorithm had trouble segmenting the thin contours of people from the noise generated during dynamic video playback. Reliability was also problematic, especially when a person occluded another person.

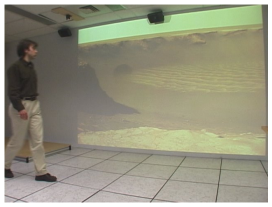
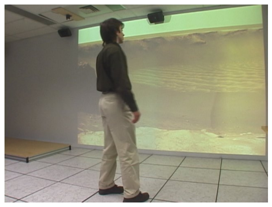
Nevertheless, when the blob tracker functioned correctly, the procedure described in Section 3.2 provided 3D information, at least as accurate as the calibration described in the previous section. Although its performance was not tested thoroughly, the precision of the tracker was not better than 5% of the image dimensions, as the bounding box often moved even when a person stood still. Also, it appears that since the same calibration information was used for the tracker and shadow removal, the large calibration errors described in the previous section did not affect as greatly as anticipated the accuracy of tracking when used in conjunction with our shadow removal module. One can appreciate this fact as well as the behavior of the tracker from the results described in the following section.

Additionally, our recent experiments with the Tyzx DeepSea V2 Stereo Camera show that stereo algorithms are capable of successfully segmenting the display surface from objects moving in front, thus achieving very reliable tracking.

5.3 Shadow Removal

Typical results of shadow removal are shown in Figures 5–2 and 5–3, for a static and dynamic video projection, respectively.[1] Although the system currently achieves reasonable results in the case of static images, some additional refinement is necessary for it to perform well with dynamic video projection. In the case of static images, we could compensate for calibration and tracker imprecision and inaccuracies by appropriately padding the tracker rectangles, increasing the dimensions by 20%. However, in the case of video projection, the tracker frequently fails, resulting in large visible shadows such as those seen in Figure 5–3(g).

[1] The full video is available from `http://cim.mcgill.ca/~saudet/research/procams2007.mp4` .


(a) frame 100

(b) frame 150

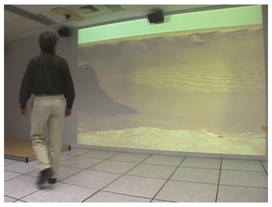
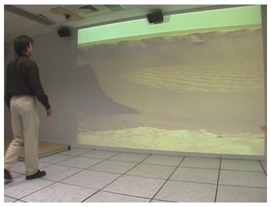
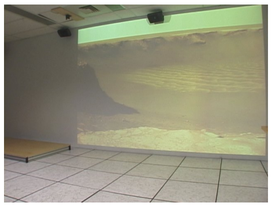
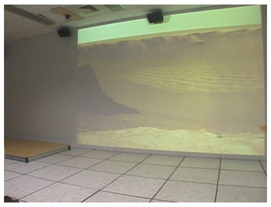
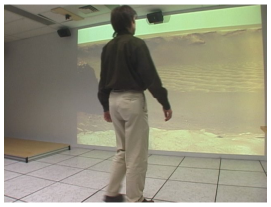
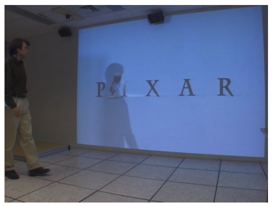
(c) frame 200

(d) frame 250

(e) frame 300

(f) frame 350

(g) frame 400

(h) frame 450

Figure 5–2: Frames from demo video with static image projection.



(a) frame 150

(b) frame 250

(c) frame 350

(d) frame 850

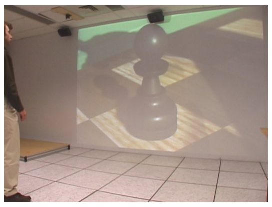
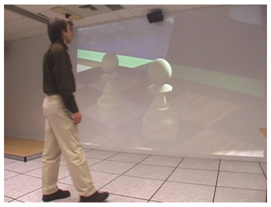
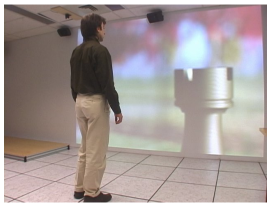
(e) frame 900

(f) frame 1100

(g) frame 1150

(h) frame 1200

Figure 5–3: Frames from demo video with dynamic video projection.

On the other hand, using the Tyzx DeepSea V2 Stereo Camera, the results are significantly improved, especially in the case of dynamic video playback.[2]

Still, an additional problem is that the system also generates artefacts during occluder movement, such as in Figure 5–3(b), because it does not display the masks in a synchronized manner. Currently, each machine receives a mask, processes it, and sends it for display without regard for whether other machines are ready to display. Obviously, synchronization would improve the results, but this entails significant software complexity [26], which we did not pursue further. Also, we hypothesize that even with a proper software implementation, synchronization of projectors at the hardware level might be required to obtain imperceptible transitions. Other visible artefacts, such as the ones seen in Figures 5–2(a) and 5–2(b), are due mostly to fast movements not correctly predicted by the Kalman filter.

Finally, since one of our objectives was to implement a system that could perform shadow removal in real time on commodity hardware, we tested the performance of the disparity contours, tracking, shadow removal, and image rectification modules on an Intel Pentium 4 2.60 GHz CPU with an NVIDIA GeForce FX 5200 graphics card. The test video contained only one person walking in front of the display surface. With the current largely unoptimized codebase,

[2] Demo with dynamic video playback:
`http://cim.mcgill.ca/~saudet/research/video_demo.mp4`
Technical demo: `http://cim.mcgill.ca/~saudet/research/tech_demo.mp4`

we obtained the results of Table 5–6, which do not take into account delays in capture, display or network transfer.

When using the higher resolutions detailed in Table 5–6, the total latency of 307 ms is too high to provide a quick enough response, but with lower resolutions, it drops to 79.3 ms. We used the latter values to obtain the results described previously in this section, and this shows that our system can perform in real time on normal hardware. This concludes our test results, whose implications we discuss in the next chapter.

Table 5–6: Delay (milliseconds ± standard deviation) for processing disparity contours, OpenCV blob tracking, shadow removal, and image rectification in OpenGL, at two different resolutions for each module, using the test video with one person. Note: Image rectification is done in parallel on multiple computers, one for each projector.

Module	Resolution	Delay (ms)	Resolution	Delay (ms)
Disparity contours	1024×768	44.5±0.5	512×384	11.3±0.2
Tracking	512×384	91.7±8.4	256×192	24.1±3.1
Shadow removal	1280×1024	159±11	640×512	39.6±3.2
Image rectification	1280×1024	11.8±2.6	640×512	4.33±1.15
Total		307±23		79.3±7.7

CHAPTER 6
Discussion

Considering the results we obtained as described in the previous chapter, compared to others, our object tracking method for shadow removal has several pros and cons. First, shadow detection [6, 18, 22, 23, 34] requires that at least one camera be placed strategically to have an unoccluded view of the entire scene. In practice, this can sometimes be challenging. Infrared occlusion detection [13, 42] requires one camera alongside each projector, in addition to the installation of near infrared floodlights at the display surface. In contrast, our method only requires that two cameras be placed appropriately for tracking people, which does not necessarily require an unoccluded view of the scene. Second, using shadow detection one cannot remove shadows before they appear. The simple approach of infrared occlusion detection effectively solves this problem by preemptively dilating the occluded region to account for possible movement in any direction. Our method goes one step further and uses prediction information from a Kalman filter, although a more carefully thought out implementation could more closely follow the movements of a person than the current results show. Because tracking can take advantage of such temporal information to predict motion, shadows can be removed before they occur. Third, since both our method and infrared occlusion detection methods do not make use of projected images, it does not matter, in principle, whether we project static images or dynamic video, and can work even in the presence of video projection at high frame rates. On the other hand, shadow detection becomes dramatically more complex in the case of video projection. Although Jaynes *et al.* [22] achieved performance of nine frames per second, it is uncertain whether this can be increased easily to 60 frames per second, the threshold at which humans stop perceiving flickering [15]. Flagg *et al.* [13] found that the combined latency of their high speed camera and projector was in the order of 50 ms, in addition to about 3 ms of processing on GPU. With this hardware and assuming comparable processing delay, the algorithm as developed by Jaynes *et al.* [22] cannot scale beyond 19 frames per second, which would of course improve with more responsive technology. However, their method still requires an unoccluded view of the whole display surface and cannot tolerate occlusion of any camera.

Our method has drawbacks as well. The calibration required for both the shadow detection and infrared occlusion detection methods can be automated relatively easily, as explained in the literature review of Section 1.3. In our case, the process is more elaborate, and due to simplifications in the implementation of our prototype, a tedious calibration phase is presently required. Furthermore, camera-based tracking requires a sufficiently illuminated environment to make the occluders visible. Even so, the simple tracking we used does not model limbs, and as such, effective shadow removal cannot be attained if the users wave their hands in front of the display. Moreover, tracking fails when one person occludes another in the camera view. In this regard, shadow detection and infrared occlusion detection methods are superior. Fortunately, our approach is not bound to any particular tracker, so these limitations may be overcome by using better or simply more appropriate tracking methods, as our recent experiments with the Tyzx DeepSea V2 Stereo Camera proves.

In our implementation, we identified various sources of calibration and tracking errors. We found that the former may result from improperly modeled nonlinear distortion in the periphery of the camera image, uneven floor or wall, and corrective homographies used for the projectors, which currently do not attempt to minimize geometric errors. However, even if we manage to fix all these issues, the current method still requires a laborious manual calibration and can only work on flat display surfaces. For these reasons, we plan to redesign the method so that it works on an arbitrary non-flat display surface while not requiring the manual calibration. The system should also work in real time on widely available commodity hardware. As future work, we propose a calibration procedure that would consist of a user walking around the room and otherwise simulating normal interaction with the computer. Using color cameras, the system would then detect shadows occurring on the display surface, similar to the way current shadow detection methods work. Our hypothesis is that without further user intervention it is possible to design a system that will map the tracking information to the detected shadows. After this automated calibration phase, the system will then be able to use the live tracking information with the mapping to locate and remove shadows from any non-flat display surface.

In the case of the tracker, although it performs well for static images, we do not obtain acceptable results with a dynamic video background. At present, it uses foreground information, the disparity contours, based solely on geometric properties of the cameras. However, we noted that in the area of the display surface, even though the disparity map was accurate, using it produced too much noise, probably caused by intrinsic differences between the two camera sensors. One way to make tracking insensitive to projector light would be to use near infrared cameras, but we do not believe them to be required, unless a dark environment is desired. A photometric color calibration should provide as good or even better performance with color cameras. There also exist better, but more complex camera-based tracking technology, such as the Tyzx DeepSea V2 System [49] that not only works well with video projection, but can cope with occlusion as well.

The work presented here provides an initial proof of concept that shadow removal can be performed on today's hardware using conventional object tracking rather than requiring the more elaborate configuration of the near infrared occlusion detection approach or other more expensive alternatives not using commodity hardware. While there remains considerable effort ahead to refine and optimize our method, current results clearly demonstrate its feasibility. We hope that this will lead to better projector-camera systems now that we have shown that we can leverage object tracking technology already in place in many front projection environments, and achieve effective shadow removal.

References

[1] Mark Ashdown and Yoichi Sato. Steerable Projector Calibration. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05) - Workshops (ProCams 2005)*, volume 3, page 98. IEEE Computer Society, 2005.

[2] Samuel Audet and Jeremy R. Cooperstock. Shadow Removal in Front Projection Environments Using Object Tracking. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007) - Workshops (ProCams 2007)*. IEEE Computer Society, 2007.

[3] Oliver Bimber, Gordon Wetzstein, Andreas Emmerling, and Christian Nitschke. 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. IEEE Computer Society, 2005.

[4] Gunilla Borgefors. Distance Transformations in Digital Images. *Computer Vision, Graphics and Image Processing*, 34(3):344–371, 1986.

[5] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab, 2006. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[6] Tat-Jen Cham, James M. Rehg, Rahul Sukthankar, and Gita Sukthankar. Shadow Elimination and Occluder Light Suppression for Multi-Projector Displays. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, volume 2, pages 513–520. IEEE Computer Society, 2003.

[7] Trista P. Chen, Horst Haussecker, Alexander Bovyrin, Roman Belenov, Konstantin Rodyushkin, Alexander Kuranov, and Victor Eruhimov. Computer Vision Workload Analysis: Case Study of Video Surveillance Systems. *Intel Technology Journal*, 9(2):109–118, May 2005.

[8] Jeremy R. Cooperstock. Interacting in Shared Reality. In *HCI International 2005: Conference on Human-Computer Interaction*, July 2005.

[9] Daniel Cotting, Martin Naef, Markus Gross, and Henry Fuchs. 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. IEEE Computer Society, 2004.

[10] Carolina Cruz-Neira. The Emerging Technology of Virtual Environments. *Journal of Aerospace Computing, Information, and Communication*, 2(2):120–124, February 2005.

[11] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *20th Annual Conference on Computer graphics and Interactive Techniques (SIGGRAPH '93)*, pages 135–142. ACM Press, 1993.

[12] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Communications of the ACM*, 35(6):64–72, 1992.

[13] Matthew Flagg, Jay Summet, and James M. Rehg. Improving the Speed of Virtual Rear Projection: A GPU-Centric Architecture. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05) - Workshops (ProCams 2005)*, volume 3, page 105. IEEE Computer Society, 2005.

[14] Matthew Flagg, Jay Summet, Ramswaroop Somani, James M. Rehg, Rahul Sukthankar, and Tat-Jen Cham. Shadow Elimination and Occluder Light Suppression for Switched Multi-Projector Displays. In *Ninth Internationl Conference on Computer Vision (ICCV '03)*. IEEE Computer Society, 2003.

[15] Yves Galifret. Visual persistence and cinema? *Comptes Rendus Biologies*, 329(5-6):369–385, May-June 2006.

[16] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[17] Maria Nadia Hilario. Occlusion Detection in Front Projection Environments Based on Camera-Projector Calirbation. Master's thesis, Department of Electrical and Computer Engineering, McGill University, 2005.

[18] Maria Nadia Hilario and Jeremy R. Cooperstock. Occlusion Detection for Front-Projected Interactive Displays. In *Advances in Pervasive Computing.* Austrian Computer Society (OCG), 2004.

[19] ITU Radiocommunication Sector Recommendation BT.709, 1211 Geneva 20, Switzerland. *Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange*, 1990. [Formerly CCIR Rec. 709].

[20] Yuri Ivanov, Aaron Bobick, and John Liu. Fast lighting independent background subtraction. *International Journal of Computer Vision*, 37(2):199–207, 2000.

[21] Christopher Jaynes, Brent Seales, Kenneth Calvert, Zongming Fei, , and James Griffioen. The Metaverse: A networked collection of inexpensive, self-configuring, immersive environments. In *7th International Workshop on Immersive Projection Technology and 9th Eurographics Workshop on Virtual Environments (IPT/EGVE '03)*, pages 115–124. ACM Press, 2003.

[22] Christopher Jaynes, Stephen Webb, and R. Matt Steele. Camera-Based Detection and Removal of Shadows from Interactive Multiprojector Displays. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):290–301, 2004.

[23] Christopher Jaynes, Stephen Webb, R. Matt Steele, Michael Brown, and Brent Seales. Dynamic Shadow Removal from Front Projection Displays. In *2001 Conference on Visualization (VIS '01)*, pages 175–182. IEEE Computer Society, 2001.

[24] P. KaewTraKulPong and R. Bowden. An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. In *2nd European Workshop on Advanced Video Based Surveillance Systems (AVBS01).* Kluwer Academic Publishers, 2001.

[25] Attila Licsar and Tamas Sziranyi. Hand Gesture Recognition in Camera-Projector System. In *Computer Vision in Human-Computer Interaction: ECCV 2004 Workshop on Human-Computer Interaction (CVHCI04)*, pages 83–93. Springer, 2004.

[26] Ikuhisa Mitsugami, Norimichi Ukita, and Masatsugu Kidode. Displaying a Moving Image By Multiple Steerable Projectors. In *2007 IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition (CVPR 2007) - Workshops (ProCams 2007). IEEE Computer Society, 2007.

[27] C. Pinhanez, F. Kjeldsen, A. Levas, G.S. Pingali, J. Hartman, A. Levas, M.E. Podlaseck, V. Kwatra, and P.B. Chou. Transforming Surfaces into Touch-Screens. Technical Report RC22273 (W0112-016), IBM Research, December 2001.

[28] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics (Texts in Applied Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, second edition, 2006.

[29] Ramesh Raskar and Paul Beardsley. A Self-Correcting Projector. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, volume 2, pages 504–508. IEEE Computer Society, 2001.

[30] Ramesh Raskar, Michael S. Brown, Ruigang Yang, Wei-Chao Chen, Greg Welch, Herman Towles, Brent Seales, and Henry Fuchs. Multi-Projector Displays Using Camera-Based Registration. In *IEEE Conference on Visualization '99 (VIS '99)*, pages 161–168. IEEE Computer Society Press, 1999.

[31] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays. In *25th Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 179–188. ACM Press, 1998.

[32] Ramesh Raskar, Greg Welch, and Henry Fuchs. Seamless Projection Overlaps Using Image Warping and Intensity Blending. In *Fourth International Conference on Virtual Systems and Multimedia (VSMM '98)*, November 1998.

[33] Yoichi Sato, Yoshinori Kobayashi, and Hideki Koike. Fast Tracking of Hands and Fingertips in Infrared Images for Augmented Desk Interface. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000 (FG '00)*, page 462. IEEE Computer Society, 2000.

[34] Rahul Sukthankar, Tat-Jen Cham, and Gita Sukthankar. Dynamic Shadow Elimination for Multi-Projector Displays. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume 2, pages 151–157. IEEE Computer Society, 2001.

[35] Jay Summet, Gregory D. Abowd, Gregory M. Corso, and James M. Rehg. Virtual Rear Projection: Do Shadows Matter? In *CHI '05 extended abstracts on Human factors in computing systems*, pages 1997–2000. ACM Press, 2005.

[36] Jay Summet, Mathew Flagg, James M. Rehg, Gregory M. Corso, and Gregory D. Abowd. Increasing the Usability of Virtual Rear Projection Displays. In *IEEE International Workshop on Projector-Camera Systems 2003 (ProCams 2003)*. IEEE Computer Society, 2003.

[37] Jay Summet, Matthew Flagg, Tat-Jen Cham, James M. Rehg, and Rahul Sukthankar. Shadow Elimination and Blinding Light Suppression for Interactive Projected Displays. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):508–517, 2007.

[38] Wei Sun. *Multi-camera object segmentation in dynamically textured scenes using disparity contours*. PhD thesis, Department of Electrical and Computer Engineering, McGill University, 2006.

[39] Wei Sun and Jeremy R. Cooperstock. An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques. *Machine Vision Application*, 17(1):51–67, 2006.

[40] Cisco Systems. Annual Report 2007, Letter To Our Shareholders, 2007. `http://www.cisco.com/web/about/ac49/ac20/ac19/ar2007/letter_to_shareholders/`, page viewed November 2007.

[41] Naoya Takao, Jianbo Shi, and Simon Baker. Tele-Graffiti: A Camera-Projector Based Remote Sketching System with Hand-Based User Interface and Automatic Session Summarization. *International Journal of Computer Vision*, 53(2):115–133, July 2003.

[42] Desney S. Tan and Randy Pausch. Pre-emptive Shadows: Eliminating the Blinding Light from Projectors. In *CHI 2002 Conference on Human Factors in Computing Systems - Extended Abstracts*, pages 682–683. ACM Press, 2002.

[43] Jean-Philippe Tardif, Sébastien Roy, and Martin Trudeau. Multi-projectors for arbitrary surfaces without explicit calibration nor reconstruction. In *Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM '03)*, pages 217–224. IEEE Computer Society, 2003.

[44] Bill Triggs. Autocalibration from Planar Scenes. In *5th European Conference on Computer Vision (ECCV '98)*, volume I, pages 89–105. Springer-Verlag, 1998.

[45] Jeroen van Baar, Thomas Willwacher, Srinivas Rao, and Ramesh Raskar. Seamless Multi-Projector Display on Curved Screens. In *7th International Workshop on Immersive Projection Technology and 9th Eurographics Workshop on Virtual Environments (IPT/EGVE '03)*, pages 281–286. ACM Press, 2003.

[46] Christian von Hardenberg and François Bérard. Bare-Hand Human-Computer Interaction. In *2001 Workshop on Perceptive User Interfaces (PUI '01)*, pages 1–8. ACM Press, 2001.

[47] Peter Weiss. Deep Vision - When walls become doors into virtual worlds. *Science News*, 161(22):344–345, June 1, 2002.

[48] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hills, July 2006.

[49] John Iselin Woodfill, Gaile Gordon, Dave Jurasek, Terrance Brown, and Ron Buck. The Tyzx DeepSea G2 Vision System, A Taskable, Embedded Stereo Camera. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '06)*, page 126. IEEE Computer Society, 2006.

[50] Mike Wozniewski, Zack Settel, and Jeremy R. Cooperstock. A framework for immersive spatial audio performance. In *The 2006 International Conference on New Interfaces for Musical Expression (NIME '06)*, pages 144–149. IRCAM - Centre Pompidou, 2006.

[51] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.