Augmentation In Visual Reality (avr)

Yunjun Zhang

University of Central Florida

Part of the Computer Sciences Commons, and the Engineering Commons

Find similar works at: https://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
https://stars.library.ucf.edu/etd/3427
AUGMENTATION IN VISUAL REALITY (AVR)

by

YUNJUN ZHANG
B.S. Tsinghua University, 1999
M.S. University of Central Florida, 2001

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2007

Major Professors: Charles E. Hughes
© 2007 Yunjun Zhang
ABSTRACT

Human eyes, as the organs for sensing light and processing visual information, enable us to see the real world. Though invaluable, they give us no way to “edit” the received visual stream or to “switch” to a different channel. The invention of motion pictures and computer technologies in the last century enables us to add an extra layer of modifications between the real world and our eyes. There are two major approaches to modifications that we consider here, offline augmentation and online augmentation. The movie industry has pushed offline augmentation to an extreme level; audiences can experience visual surprises that they have never seen in their real lives, even though it may take a few months or years for the production of the special visual effects. On the other hand, online augmentation requires that modifications be performed in real time. This dissertation addresses problems in both offline and online augmentation.

The first offline problem addressed here is the generation of plausible video sequences after removing relatively large objects from the original videos. In order to maintain temporal coherence among the frames, a motion layer segmentation method is applied. From this, a set of synthesized layers is generated by applying motion compensation and a region completion algorithm. Finally, a plausibly realistic new video, in which the selected object is removed, is rendered given the synthesized layers and the motion parameters.
The second problem we address is to construct a blue screen key for video synthesis or blending for Mixed Reality (MR) applications. As a well researched area, blue screen keying extracts a range of colors, typically in the blue spectrum, from a captured video sequence to enable the compositing of multiple image sources. Under ideal conditions with uniform lighting and background color, a high quality key can be generated through commercial products, even in real time. However, a Mixed Reality application typically involves a head-mounted display (HMD) with poor camera quality. This in turn requires the keying algorithm to be robust in the presence of noise. We conduct a three stage keying algorithm to reduce the noise in the key output. First a standard blue screen keying algorithm is applied to the input to get a noisy key; second the image gradient information and the corresponding region are compared with the result in the first step to remove noise in the blue screen area; and finally a matting approach is applied on the boundary of the key to improve the key quality.

Another offline problem we address in this dissertation is the acquisition of correct transformation between the different coordinate frames in a Mixed Reality (MR) application. Typically an MR system includes at least one tracking system. Therefore the 3D coordinate frames that need to be considered include the cameras, the tracker, the tracker system and a world. Accurately deriving the transformation between the head-mounted display camera and the affixed 6-DOF tracker is critical for mixed reality applications. This transformation brings the HMD cameras into the tracking coordinate frame, which in turn overlaps with a virtual coordinate frame to create a plausible mixed
visual experience. We carry out a non-linear optimization method to recover the camera-tracker transformation with respect to the image reprojection error.

For online applications, we address a problem to extend the luminance range in mixed reality environments. We achieve this by introducing Enhanced Dynamic Range Video, a technique based on differing brightness settings for each eye of a video see-through head mounted display (HMD). We first construct a Video-Driven Time-Stamped Ball Cloud (VDTSBC), which serves as a guideline and a means to store temporal color information for stereo image registration. With the assistance of the VDTSBC, we register each pair of stereo images, taking into account confounding issues of occlusion occurring within one eye but not the other. Finally, we apply luminance enhancement on the registered image pairs to generate an Enhanced Dynamic Range Video.
Dedicated to my parents and my wife
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my academic advisor, Dr. Charles E. Hughes, for his guidance, support and consideration during my years of graduate study. It would have been impossible for me to finish my Ph.D research without his direction. On top of academic advice, Dr. Hughes brightens my view towards life.

I would like to thank Dr. Xin Li, Dr. J. Michael Moshell and Dr. Ratan K. Guha, who as my committee members have given me significant and valued advice on my research. Also I would like to thank my previous advisor, Dr. Mubarak Shah, for introducing me to the wonderful world of research. I would like to thank Dr. Kazumasa Yamazawa for his help on all types of questions that I can think about.

During this long, pressure-filled time of working on my Ph.D, I have received countless unselfish help from my previous and current fellow lab-mates, especially Jiangjian, Yun, Yuping, Arslan, Paul, Xiaochun, Mark, and Elena. I will always cherish the time when a short discussion inspired a great idea.

I would like to especially thank my parents for their unconditional love and constant spiritual and financial support over all these years.
Finally, I am deeply grateful to my beloved wife, Donghang, who is always by my side. This is a moment that I feel so sorry about my dry language since I cannot find a proper sentence to express my love for her.

I would like to recognize the support provided to this research effort by the Office of Naval Research's Virtual Technologies and Environments (VIRTE) program.
TABLE OF CONTENTS

LIST OF FIGURES ...................................................... xiii

LIST OF ACRONYMS ................................................... xxii

CHAPTER 1 INTRODUCTION ............................................. 1

CHAPTER 2 RELATED WORK ........................................... 11
  2.1 Video Completion and Inpainting ................................. 11
  2.2 Blue Screen Keying and Matting ................................. 13
  2.3 Camera-Tracker Calibrations .................................... 16
  2.4 HDR imaging .................................................... 20

CHAPTER 3 VIDEO COMPLETION ................................... 23
  3.1 Object Removal in Videos ........................................ 24
    3.1.1 Motion Layer Extraction .................................. 26
    3.1.2 Layer Compensation and Completion ....................... 28

ix
CHAPTER 4 NOISE REDUCTION IN BLUE SCREEN KEYING

4.1 Introduction

4.2 Edge Based Tri-Map Construction and Keying

4.2.1 Chroma Keying Algorithm

4.2.2 Generating Key Boundary Map with Confidence and Constructing Tri-Map

4.2.3 Algorithm Evaluation

4.3 Refined Blue Screen Keying

4.3.1 Anisotropic Image Filtering

4.3.2 Hybrid Seeded Region Growing

4.4 Results and Comparisons

4.5 Conclusion and Future Work

CHAPTER 5 CAMERA-TRACKER CALIBRATION

5.1 Introduction

5.2 Naive Camera-Tracker Calibration
LIST OF FIGURES

Figure 1.1  A snapshot in the silent movie “A trip to the moon” (1902). 2

Figure 1.2  Overview for augmentation in visual reality. Virtual objects augment the original scene. Online augmentation enables interaction and thus requires real time video refresh rate. 3

Figure 2.1  Move the camera-tracker rig from position 1 to position 2. $A$ can be calculated as $A = A_2 A_1^{-1}$ and $B$ can be calculated by $B = B_2 B_1^{-1}$. Note that the direction of the array in the figure is critical for the matrix calculation. 16

Figure 3.1  (a) One frame of the original video. (b) The result of the motion layer extraction. The layers are numbered based on their overlapping order. (c) and (e) All the layers except the one we want to remove. (d) and (f) Layers synthesized by applying motion information of all the frames in the video. (g) Pixels which are still unknown are filled with our region completion method. Bottom: Selected frames from the original and the synthesized video. 25

Figure 3.2  The final motion segmentation results of two frames in mobile-calendar sequence. The red pixels are the occluded ones. 26
Figure 3.3 This is the demonstration of the three layers in mobile-calendar sequences. The order of the layers is from left to right. In the real frame, the ball and the train belong to the third layer; the calendar is the second layer and the background is the first layer.

Figure 3.4 Left column: Layers with missing regions. Right column: The region completion results obtained by our method.

Figure 3.5 Left Top: A target frame in which a large region of the car layer is covered by the map board. Right Top: The reference frame in which the car layer is fully visible. Left Bottom: Synthesized car layer by warping the car images based on the directly calculated motion parameters. Right Bottom: Final result of our method by applying the constant velocity constraint.

Figure 3.6 Top row: Five selected frames from the original mobile-calendar sequence. Bottom row: The correspondent frames obtained by our algorithm, in which the train and the ball are removed.

Figure 3.7 Top row: Five selected frames from the original car-board sequence. Bottom row: The corresponding frames obtained by our algorithm, in which the map board is removed.

Figure 3.8 Top row: Five selected frames from the original statue sequence. Bottom row: The corresponding frames obtained by our algorithm, in which the statue is removed.
Figure 4.1  Scheme overview of the algorithms we proposed in this chapter. The extended parts are colored in red. .......................................................... 41

Figure 4.2  (a) A noisy input image taken by our head mounted display camera. (b) Key result by our method. (c) Composition result with a white background. ........ 43

Figure 4.3  (a) The original input image. (b) Key mask result based on our described PCA-based algorithm. (c) Canny Edge detection result. (d) Boundary map based on the key output. (e) Boundary map with the edge agreement confidence. (f) Tri-Map generated by the confidence boundary map. (g) Compositing result based on the PCA key over a green background. (h) Compositing result of our algorithm, note that the noise in (g) are all successfully removed. (i) One more compositing result over the background in our lab. ......................... 49
Figure 4.4  (a) The original input image. (b) Key mask result based on the plane cut algorithm. A large portion of the background is classified wrong. (c) Canny Edge detection result. Due the the similarity of the foreground color in the trouser’s area to the background color, the detected edges are not located closely on the actual foreground boundary. (d) Boundary map based on the key output. (e) Boundary map with the edge agreement confidence. Part of the boundary has 0 confidence. That causes the boundary to be broken into non-connected pieces (f) key output by the boundary confidence map. Since the boundary is not closed, the morphological image filling algorithm cannot be used to construct a correct tri-map. 50

Figure 4.5  Two filtered images of the input from Figure 4.3 and Figure 4.4 by the fast bilateral filtering algorithm. Most of the camera noise is reduced or removed perceptually. 52

Figure 4.6  Left: The initial seed region for Figure 4.3 acquired by calculating $2B - G - R > 0.6$. This constraint is so strong that only a few pixels remain in the background region as seeds. The color is inverted for better visualization. Right: With the limited seeds, our SGR algorithm correctly grows the background into the proper area. 54
Figure 4.7 A demonstration for balanced and unbalanced neighborhood area. In the balanced case, both $\Pi_u$ and $\Pi_d$ are noisy but the average colors represent the potential growing possibility. In the unbalanced case, only one noisy pixel has its color away from the background region. However, it contributes a lot for the average color in $\Pi_u$ that may stop the growing in this case.

Figure 4.8 One frame result to compare the growing conditions. (a) SRG by only the color constraint without balance factor. Note that the background region grows into the trouser's area since the neighborhood color in that area is similar. Without the balance factor, many isolated regions cannot be correctly grown into. (b) SRG by only edge constraint. As the edge result in (d) shows, a small portion of the left bottom in the foreground is not connected well. Therefore this edge only result grows the background into the foreground area. (c) SRG by hybrid growing. The boundary of the foreground area is properly maintained.

Figure 4.9 (a) A synthesis image with pure blue background. Both the background and foreground regions are known as ground truth. (b) Image result by adding zero mean Gaussian noise with variation 0.12 to (a). (c) Plane cut result for $2G - B - R > 1.9$. By this high threshold the foreground region is free of wrong classifications. (d) Keying result by our algorithm. (e) It shows that when the noise variation increases, our algorithm achieves accuracy rates in the foreground and background close to 1 and remains constant.
Figure 4.10 A compositing sequence with blue background. The image on the top is one of the original frames, while the other six images are selected compositing frames.

Figure 4.11 A compositing sequence with green background. The image on the top is one of the original frames, while the other six images are selected compositing frames.

Figure 4.12 This figure shows the results of one of the hardest blue screen inputs. The original frame shown in the top are taken in a low light condition, therefore noise and shadow merge together and it is very hard to distinguish them even interactively. From left to right, top to bottom: our result with the input resolution at $320 \times 240$; our result with the input resolution at $640 \times 480$; our real time result described in section 4.2.1; result by plane cutting algorithm from section 4.2.1; result based on Vahols’ patent; result by a commercial keying software called combustion.

Figure 5.1 An illustration for the possible coordinate frames existing in MR system. From the figure, it is straight forward that the transformation between physical tracker(sensor) and cameras can be recovered when the world model can be measured in tracking frame. In reality, measuring a position (LED light point) in tracking frame is much easier than measuring a full transformation between world and tracking frames.
Figure 5.2 A closeup view of our camera-tracker rig for the MR system and the testing configuration for our one point algorithm. The left image shows the LED light solidly mounted on a calibration checkerboard. The checkerboard is used for Park-Martin’s method. The right image show the actual camera-tracker rig setup. The tracker’s coordinate frame is demonstrated in red color and the camera frame is in blue.

Figure 5.3 Top images shows the extrinsic calibration result for our camera. The camera poses shown in the image clearly demonstrates that the way we took images can cover a good range of space for our testing purpose. Bottom image gives the coordinate frame for the checkerboard.

Figure 5.4 This plot shows the comparison between the average reprojection errors for the DLT only algorithm and the proposed approach. X-axis is the width of the offset windows $w$ and Y-axis is reprojection error.

Figure 5.5 This plot shows the comparison between the average reprojection errors for the Levenberg-Marquardt optimization and the optimization with a robust Tukey estimator.

Figure 5.6 Left image shows one of the captured images with all background light off. Only the LED is visible. Right image is from the same camera pose with all light on. Notice that the calibration checkerboard in the image is only used for PM method.
Figure 5.7  Three typical frames of the reprojection results. Blue dots in the images are our results, and the red dots are results by PM method.

Figure 6.1  Layout of our Mixed Reality System – User acquires real scene from a video see-through HMD. Scene is processed using pre-scanned virtual geometry that, while not displayed, assists in the registration of the binocular images.

Figure 6.2  Flowchart of our luminance enhancement system.

Figure 6.3  An overview of the camera coordinate frame – For a general projective camera model, the projection center is called camera center \( C \). The plane passing \( C \) and parallel to the image plane is the principal plane. The \( Z \) axis for the camera coordinate system is defined as the principal axis. The point where the principal axis meets the image plane is called the principal point \( p \). The actual image origin is defined as \( O \).

Figure 6.4  The sparse 3D point cloud is projected onto the left and right images. Each point in the cloud represents a cross on the checkerboard. A radial distortion model is applied to achieve accurate registration.

Figure 6.5  Left image is actual input from the left camera with a low brightness setting. Right image is the input from the right camera with a high brightness setting.
Figure 6.6  In this sequence, a paper box is attached on the wall to demonstrate the advantage of our method. (a) Left input with the projected point cloud on it. (b) One frame of the enhanced result of our method. (c)-(f) Left: two frames of the our enhanced results. Right: the corresponding direct merged results. (g) A frame of the enhanced result with indoor and outdoor mixed view. (h) Left input. (i) Right input.

Figure 6.7  One view of the point cloud model for our pinball game machine. The three bright boards around machine are markers used for model registration.

Figure 6.8  (a) Left input. (b) Right input. (c) Disparity map generated by VDTSBC. Note that only the checker board area is modeled. (d) Directly merged result for the left display. The displacement is large due to the close camera position. (e) Merged left result of our method. The brighter checkerboard is achieved by combining the checkerboard from right image to the entire left image. (f) Result of the checker board only.

Figure 6.9  From left to right: three typical enhanced frames for the modeled game table on the top and the corresponding direct merged results in the bottom.
LIST OF ACRONYMS

AR Augmented Reality

AVR Augmentation in Visual Reality

DOF Degree of Freedom

HMD Head Mounted Display

LMS Local Motion Scale

MR Mixed Reality

NPR Non-Photorealistic Rendering

SIFT Scale-invariant Feature Transform

SRG Seeded Region Growing

VDTSBC Video-Driven Time-Stamped Ball Cloud

VR Virtual Reality
CHAPTER 1
INTRODUCTION

In this dissertation, we introduce the term Augmented Visual Reality (AVR) to denote any experience that includes the modification of human visual perception. Offline AVR refers to operations that modify the visual input in non-realtime. For this class of applications the augmented output is pre-constructed and can be delivered to the end users when needed. On the other hand, online AVR aims at realtime performance and user interaction. Both of these categories present the end users altered visual information, providing new means by which people can visually perceive the world.

Inspired by the desire to create a new form of entertainment, pioneers in the motion picture industry augmented their raw shots with visual effects. Figure 1.1 presents a frame of an early silent film “A trip to the moon” (1902), which clearly demonstrates a scene that does not exist in the real world. Television, an invention first commercialized in the 1930s, has brought motion pictures into almost every family’s living room. Now, most of us take what we watch on the TV or at the movies for granted, not dwelling on the fact that a large portion of the content is due to the augmentation of visual reality. Nonetheless, all motion pictures share a common property: they do not involve individual user interactions, except for some trivial exceptions, e.g., those involving choices of endings. This non-interactivity gives the video makers a chance to augment the content.
offline on the two dimensional representation without worrying about variations in the audience members’ points of view.

Although the movie-TV overlay technique has been highly successful, its lack of user interaction has motivated the development of new forms of media. The focus of this dissertation is on computer-mediated visually enhanced experiences, addressing research problems related to systems typically classified as virtual, augmented and mixed reality.

In order to simplify the taxonomy employed in this dissertation, we use mixed reality to refer to any combination of physical and virtual reality. With this simplification, existing interactive visual systems can be classified into two categories: VR and MR. In a VR system, all the visual content the user sees is synthetically generated. The user either views the images non-immersively by monitor, projection screen or dome screen, or views them immersively wearing a head mounted display. In an MR system, the user can see both virtual and real objects at the same time. As such,
the visual content must be properly registered and the user needs to be enabled to interact in real
time with both virtual and real objects.

Figure 1.2 gives an overview for both offline and online augmentation in visual reality.

A common task in AVR is object removal and insertion. Video overlay is a typical object inser-
tion application. It refers to the case where new content is inserted over an existing background. If
the main objective is to maintain the consistency of the video scene after removing a portion of the
image, object removal techniques come into play. For example, for a historical drama show, if the
raw video has a modern motor vehicle in the view and retaking the video is not a option, the only
choice left is to remove the vehicle and repair the hole with appropriate background information.
The motivation for object removal and insertion can be traced back to the Renaissance [BSC00].
The practice was to restore the deteriorated portions of historical artwork in a manner that main-
tains believability. This motivation was naturally extended in the last few decades for augmentation of photographs and motion film. The ultimate requirements for these applications are:

- The geometrical information for the augmented part needs to match the original video.
- The consistency of the lighting condition needs to be kept in the modified part.
- The boundary of the augmentation requires a natural merging mechanism.

A related form of visual augmentation requires that those parts of the image designated as foreground be retained, while the background is replaced or augmented with other visual content. Blue screen keying [SB96], which puts a constant background color (normally blue or green) behind a foreground, is a common approach to extract objects and characters of interest that can be smoothly blended with a new background. Theoretically, the blue screen keying problem can be correctly solved by taking pictures of the same foreground against two known constant background colors. In reality, though, the problem does not require such a complex setup and can be efficiently and effectively solved using heuristics [SB96]. For instance, [SHM02] construct a MR simulation system in which a blue screen keying algorithm is applied online to remove the visual appearance of designated backgrounds for virtual augmentation, e.g., blue screens in windows and doors are replaced by virtual models appearing to be inside physical buildings. Unfortunately, the noise level on the video see-through HMD cameras, and the color variations for the background due to lighting and material inconsistency result in an inconsistent spotty matte. These limitations inspire
the research presented here to improve keying quality in noisy contexts, while retaining real-time performance.

Once a keying algorithm isolates the foreground from the background, new content can be added to meet an application’s requirements. However, this is not always a straightforward task due to the required coherence between the projection of the foreground and that of the added background. For example, combining a foreground taken by a side view camera with a background captured by a top view moving camera may generate a visually unacceptable scene. To compensate for this, the movie industry has adopted vision-based camera pose tracking. By recognizing fixed 3D features in a given scene, a relative coordinate frame of the world can be generated and new content can be added based on this frame of reference. The accuracy of such a system can be sub-pixel with motion compensation being used to meet the consistency constraint.

For online AVR systems, the coherence problem remains a challenge. Mixed reality applications, as practical realtime AVR systems, have undergone rapid development with the help of computer graphics research. As a consequence, rendering and capturing video at appropriate speeds is no longer an issue. However, merging virtual and real objects in a plausible manner is still not a straightforward task due to the alignment constraints. A significant challenge is to accurately recover the camera pose in real time. A noisy pose estimation may result in a jittery merge between the real and virtual objects, which is visually disconcerting to end users since jittering or misalignment of a portion of the scene in the video may cause misinterpretation. Also, any noticeable delay in the output of the video is a major reason for motion sickness in MR applications. For a typical
MR system with a stereo see-through HMD and pose tracking equipment, a few factors decide the accuracy of the mixed video stream.

- The quality of the tracking equipment.
- The accuracy of the stereo camera calibration.
- The accuracy of the transformation between the tracker and the cameras.
- The accuracy of the transformation between the tracking system and the user defined world coordinate frame.
- The latency between the tracker and the camera output.
- The latency between the sensed balance from the user’s vestibular system and the video stream output. This delay is caused by insufficient speed of the video capture hardware, and/or the software load for the rendering and application logic.

There is a collection of 3D pose tracking devices previously developed, which have been incorporated in existing MR systems.

**Magnetic:** With an oriented magnetic emitter and a sensor, a magnetic tracker can achieve a 6 degree of freedom (DOF) pose estimation. Though ideal conditions result in acceptable accuracy, the tracking quality suffers from magnetic field distortion caused by nearby metal conductive surfaces.
Mechanical: A predefined position is attached physically to the user and the position of the user can be accurately retrieved. However, the working volume of the user is highly limited by this setup.

Optical: A group of sources emits light onto the object so the position of the object may be recovered. These systems require a clean path between the light and the object to track. No occlusion is allowed.

Ultrasonic: A collection of ultrasonic speakers with a receiver make up the system. Due to the wave length of the sound, this approach is not as easily affected by direct occlusion as are optical trackers. However, noise level and echo may prevent the system from achieving a good pose estimation.

Inertial: Inertial trackers use a combination of micro-accelerometers and a compass to determine the orientation of the tracker. This approach can achieve a high refresh rate; however small errors are accumulated at each step, leading to an almost unavoidable drift.

Vision: Tracked 2D features in the video stream can retrieve the camera pose precisely when the camera’s internal parameters and the 3D feature location are precisely known. It normally costs less to apply vision-based methods for tracking, and the accuracy can be surprisingly good. Vision-based tracking has trouble when the camera moves too fast or there are too few features that can be tracked. Also, restarting the tracking after failure can be cumbersome.

Hybrid: This refers to any system that is a combination of two or more of the above techniques.
Theoretically, the vision-based tracking methods can give the best results. One reason for this is that, although the accuracy of the pose estimation is critical for MR applications, the minimization of the retrojection error, which is the minimization scheme for most of the vision-based pose tracking algorithms, is the ultimate goal. Unfortunately, vision-based methods either require visually obtrusive markers or are hard to be restarted if tracking gets lost. These disadvantages mitigate the practical value for applying vision-based methods to realtime MR systems.

Practically, physical based tracking provides more reliable, but arguably less accurate pose tracking with respect to the tracker unit. At the same time, these systems add more transformations between the camera and the world. The required transformations between the tracker and the camera, and between the tracking system and the world are normally fixed but not easy to measure.

The realtime requirement for MR systems imposes a limit on the data transfer rate for HMD video streams. This, in turn, limits the size and depth of the acquired images. Specifically, the displayable luminance for a typical HMD camera is low so that part of the visible scene may be saturated when a bright light source is presented. Even for indoor MR system, this may be a problem if the sun is visible through a window or door.

In this dissertation, we address these aforementioned problems in both offline and online augmentation. We start with Chapter 2, which provides an introduction to the relevant research supporting AVR.

In Chapter 3, we address one of the video editing problems described earlier, object removal. The challenge of object removal in video is to maintain spatial and temporal consistency between
the recovered region and the untouched background. We propose a solution that applies a texture-based image repairing technique on pre-segmented regions, and propagates the repaired background based on the motion parameters for each segmentation. The original overlapping order among the layers is properly maintained; therefore the acquired completion retries only the related information from designated layers.

In Chapter 4, we provide a way to construct a blue screen key for video synthesis or the blending required in Mixed Reality applications. A three-stage keying algorithm is presented to reduce the noise in the key output. First, a standard blue screen keying algorithm is applied to the input to get a noisy key; second, the image gradient information and the corresponding region are compared with the result in the first step to remove noise in the blue screen area; and finally, a matting approach is applied on the boundary of the key to improve key quality. Due to the computational demand of the current method, the presented algorithm is classified as an offline approach. However, since this procedure requires no batch information, there is no intrinsic barrier to its becoming an online technique when computational power increases to meet its demands.

In Chapter 5, we address the problem to calculate the transformation between MR head mounted camera and the affixed tracker. Accurately deriving this transformation is critical for MR applications. This transformation registers the camera’s local coordinate frame into the tracking system, which in turn registers the virtual content frame with the real tracking frame. We carry out a non-linear optimization method to recover the camera-tracker transformation with respect to the image reprojection error.
In Chapter 6, a practical luminance extension to the existing MR hardware system is discussed. For the currently commercialized stereo HMD display systems, the luminance range that each camera can cover is very limited. We address this limitation by introducing the concept of Enhanced Dynamic Range Video into the MR domain. We construct a Video-Driven Time-Stamped Ball Cloud (VDTSBC), which serves as a guideline and a means of storing temporal color information for stereo image registration. The positional data for each Ball in the VDTSBC is acquired by precise measurement, usually via a 3D terrestrial laser scanner, while the time-stamped dual-channel color parameters are projected from two corresponding images captured from the stereo rig. With the assistance of the VDTSBC, a pair of stereo images, each based on a different brightness setting, can be registered, even in the presence of areas occluded in one but not the other camera view.

In order to generate a full radiance map that can cover the human eye contrast ratio in an indoor-outdoor combined MR environment, more than two frames taken under distinct exposures are required [DM97]. This is an impractical requirement for an online MR system using the “eyes” of the HMD. To achieve capabilities similar to the human visual system’s ability to view bright sunlight and dark indoor details at the same time, we abstract the real world radiance with a low and a high partial radiance map. After registering and combining the two views, the over-saturated and under-illuminated regions can be represented in a single frame that merges the features extracted from each.

Chapter 7 concludes the thesis by providing a summary and directions for future research.
CHAPTER 2
RELATED WORK

Augmentation in Visual Reality is an area of human endeavor with a recorded history that goes back thousands of years. It includes theatrical techniques developed by the ancient Greeks, e.g., the use of cloth and wind to simulate fire, and artistic techniques, e.g., perspective and color temperature, developed in Asia and during the Renaissance. AVR may even go back to prehistoric times with the interaction between fire and primitive cave drawings. A complete history of AVR is well beyond the scope of this thesis and so, in this chapter, we focus only on prior research that is directly relevant to the work presented here. These areas are video completion and inpainting, blue screen keying and matting, camera-tracker calibration, and high dynamic range imaging.

2.1 Video Completion and Inpainting

Most of the previous work for missing data recovery has focused on single image completion. There are two primary categories of work in this area. One was introduced by Bertalmio [BS00], who used a PDE-based method to repair damaged images. The idea is to extend the structures around the boundaries of the damaged area, and to fill the color information properly. For an image in which only small portions are missing, this approach can achieve very smooth results. However, using their method, the lack of texture in large reconstructed areas creates an image that
is visually unacceptable due to discontinuities around the completed region. These are especially apparent in natural images. Levin et al. [LZW03] extended the idea by measuring the global image statistics, so that the inpainting results are based on prior image knowledge in addition to local color information.

Some researchers have considered texture synthesis based methods as a way to achieve image completion [BVS03, CPT03, DCY03b, JT03, IP97]. Criminisi et al. used the angle between the isophote direction and the normal direction of the local boundary to define the searching order of the patches, so that the structure of the missing region can be completed before filling in the texture [CPT03]. Jia and Tang [JT03] explicitly segmented the unknown area into different homogeneous texture areas using tensor voting. Drori et al. [DCY03b] incorporated the combination of pyramid image approximation and adaptive neighborhood size to achieve impressive results. However, this method is slow due to its high computational complexity.

Recently some researchers started to address the video repairing problem. Bornard et al. used neighborhood-frame correction to repair damaged motion pictures [BLL02]. Bertalmio also addressed video repair in [BVS03]. Wexler et al. filled the missing video portions by sampling spatio-temporal patches from other video portions, while enforcing global spatio-temporal consistency [WSI04]. Other interesting work has been done by Jia et al. [JWT04]. Most existing repairing methods do not take the motion layers and their orders in the videos into consideration. By applying the layer order, the occlusion ambiguity can be clearly identified.
2.2 Blue Screen Keying and Matting

Blue Screen techniques assume a constant background that enables a precise calculation of the transparency ($\alpha$) value. Smith and Blinn [SB96] generalize the solution for blue screen matting when given two known background colors behind a stationary foreground with the assumption that the perceived real foreground color over the two backgrounds should be the same. In the paper, the authors clarified three solvable situations for blue screen problems. The first one requires a linear relationship between the three channels of the real foreground color. With this assumption the actual color only contains 2 degrees of freedom. Together with the unknown $\alpha$ weight between the foreground and the background, 3 unknowns are embedded in 3 linear observation color channels, leading to a trivial solution. The second and the third solutions are similar where they prove that, when 2 sets of background colors (one fixed for solution 2 and both arbitrary for solution 3) are used separated behind a stationary foreground, 6 linear equations can be derived and the problem becomes over-constrained. This inspired many subsequent researchers to address image keying, but the stationary requirement limits the applicability of these approaches for videos.

Instead of knowing the constant background color, some researchers estimate the background by gathering neighbor color information. This enables natural image matting from a single shot with a random background color.

Chuang et. al. [CCS01] solve the matting problem by building foreground and background probability from a given neighborhood. They formalize the problem as a maximum a posteriori framework for the probability distribution function of foreground, background and $\alpha$ with respect
to the color observation. A log likelihood operator is then applied to linearize the equation. A linear solution can be achieved for the foreground and the background when the $\alpha$ value is fixed. The $\alpha$ value can then be updated using the calculated foreground color, $F$, and background color, $B$.

Sun et. al. [SLK06] extends the previous approach by taking additional images for the same scene with the camera flash on, assuming the background is far enough away to not be influenced by the flash. This efficiently reduces the false background estimation. At the same time the authors formulate the problem by a joint Bayesian scheme, which adaptively estimates the $\alpha$ value by altering the calculation between the no-flash image and the flash-only image.

With a fixed background, Apostoloff and Fitzgibbon [AF04] extend the Bayesian framework by applying a spatial-temporal constraint on the log-likelihood of $\alpha$. This improvement makes the video matting available. Based on the observation that the $\alpha$ values are more likely to be 0 or 1 and $\alpha$ edges are tightly correlated with edges in the composite image, the authors model the distribution for $\alpha$ as a beta distribution.

Mitsunaga et. al. [MYT95] start to use the gradient of the image in the boundary area of the foreground to simulate the change of $\alpha$ weight. This is based on an assumption that, on the boundary, the color of the foreground and the background change far smaller than the change of the weight. The actual value of $\alpha$ can then be integrated along a 1D path that is perpendicular to the boundary of the foreground. This approach only requires the difference between the foreground
and the background instead of their absolute values. Sun et al. [SJT04] extend this approach by solving a Poisson equation in the 2D image space directly.

McGuire et al. [MMP05] develop a customizable multiple camera rig in which every camera in the rig shares the same optical axis by beam-splitters. They apply the camera rig for the matting problem by setting up the system with one pinhole camera, one camera focused on the foreground, and one camera focused on the background. They simulate defocusing these images by convolution on the foreground, background, and $\alpha$ channel. This modifies the classical matting formula shown in equation 4.6 into 3 convolution equations according to the focus distance while there are still 7 unknown parameters for each pixel location. Therefore a total of 9 observed color channels over-constrains the matting problem, which now can be solved by a minimization scheme. The authors globally minimize the solution for all image locations at the same time. They claim the gradient descent minimization algorithm that they applied is better than other choices since it does not require a calculation for the pseudo-inverse of a large scale sparse Jacobian matrix. The tri-map that is generally constructed by manual means in other approaches can be evaluated automatically. The authors compare the texture frequency between the front-focused image and back-focused image to separate foreground and background. This requires the assumption that the image needs enough texture contents in both the foreground and the background area. On top of this assumption, the authors also assume the depths of the foreground and background need to be known at least at an approximate level in order to calculate the defocus radius.
Figure 2.1: Move the camera-tracker rig from position 1 to position 2. $A$ can be calculated as $A = A_2 A_1^{-1}$ and $B$ can be calculated by $B = B_2 B_1^{-1}$. Note that the direction of the array in the figure is critical for the matrix calculation.

2.3 Camera-Tracker Calibrations

The transformation of a camera and an affixed tracker unit can be expressed by a close form equation:

$$AX = XB$$ (2.1)

where $A$ and $B$ are the known intermediate transformations of the camera and the tracker, and $X$ is the unknown relation between the camera and the tracker. Figure 2.1 shows the way to acquire $A$ and $B$.

Existing research primarily focuses on the presentation of the orientation and translation. Shiu et. al. [SA89] skip the calculation for $A$ and $B$ since $A$ can be easily retrieved by a robot controller,
while $B$, as the relative transformation for a camera, can be calculated by 3D pose estimation. The homogenous matrix transformation equation 2.1 can be expressed as

$$
\begin{bmatrix}
    R_A & t_A \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    R_X & t_X \\
    0 & 1
\end{bmatrix}
= 
\begin{bmatrix}
    R_X & t_X \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    R_B & t_B \\
    0 & 1
\end{bmatrix}
$$

(2.2)

where $R$ is a $3 \times 3$ rotation matrix and $t$ is a translation vector.

This matrix equation can then be separated into a rotation only part and a translation only part

$$
R_A R_X = R_X R_B 
$$

$$
R_A t_X + t_A = R_X t_B + t_X 

(2.3)
$$

From this the authors prove that $R_A$ and $R_B$ have the same angle of rotation. Each of the separated equations has one degree of freedom, therefore equation 2.1 has 2 degrees of freedom, which implies an infinite number of solutions. In order to find a unique solution, at a minimum the camera-rig position needs to be set up so that 2 sets of equation can be generated. In the original paper, the authors show that, when the rotation axes are not parallel or anti-parallel, and the rotation angles are not 0 or $\pi$, the solution for the rotation matrix is unique. When the rotation matrix is discovered, the translation part is trivial to compute by using a least square fit solution. This initial attempt to recover Hand-Eye calibration does not consider the orthonormal constraint for the rotation matrix. Moreover, the two-step calculation propagates the calculation error in the rotation part into the translation vector. It is also easy to see that a $3 \times 3$ rotation matrix representation is redundant.
Park and Martin [PM94] introduce the Lie Group concept that defines a logarithm and an exponential operation mapping rotation matrix to a skew-symmetric matrix of the form

\[
\begin{bmatrix}
0 & -\omega_3 & \omega_2 \\
\omega_3 & 0 & -\omega_1 \\
-\omega_2 & \omega_1 & 0
\end{bmatrix} = [\omega]
\] (2.4)

By mapping the rotation matrix in $A$ and $B$ into a logarithm form, the transformation can be easily retrieved in a two-step manner similar to [SA89].

Employing vision-based approaches started as early as in [TL88a]. Basically the authors apply camera calibration with respect to a calibration board. Using this, the extrinsic camera pose can be recovered from the board frame. A quaternion is used to define the rotation matrix only. A full vision-based calibration approach is represented in [Gar99].

A quaternion-based method is proposed in [CK88]. Although the method only handles the rotation cases, it clearly shows the advantage of this representation form in comparison to a rotation matrix. Ikits [Iki00] reduces the parameterization for rotation into its true degree of freedom (3) by expressing the rotation with a reduced quaternion expression. Given a unit quaternion element of $q = [q_0, q_1, q_2, q_3]^\top$, a minimum magnitude vector component of $q$ can be defined by

\[
s = \begin{bmatrix}
q_i \\
q_j \\
q_k
\end{bmatrix}
\] (2.5)

in which the component with the largest magnitude is removed.
Daniilidis [Dan99] presented a unified representation by unit dual quaternions to express a line transformation. They consider the camera and motor transformation as screws. Using this approach, they can facilitate a simultaneous solution for the Hand-Eye transformation using singular-value decomposition (SVD).

In order to calculate $X$, the 6-DOF camera pose with respect to the predefined reference frame needs to be estimated precisely. However, it is well-known that the vision-based camera pose estimation could carry a noticeable amount of translation uncertainty along the camera axis [Hof99].

Tuceryan et. al. [TGN02] took a minimum number of predefined land markers for alignment and manually set up a collection of correspondences with 2D on-screen markers and the land markers. A minimum of 6 pairs of correspondences are required, though in practice the authors applied least squares estimations on more correspondences to get robust results. Genc et. al. [GTN02] extended the work by simplifying the virtual camera model, in which only the changing parameters are considered.

Daillot et. al. [BJB03a] designed a tracker alignment framework that relies on a single user-defined land marker to extract camera-tracker calibration. For a simple setup, they used two land markers to interactively align the optical hand-mounted camera to a known pose. Though easy for user interaction, the accuracy heavily relies on a set of user measurements. In cases where they do not have a world on which to base the transformation, they apply similar methods as in [PM94] with multiple relative pose measurements.
Some recent research extends the focus for Hand-Eye tracking. Malm and Heyden [MH00] introduced an optical flow concept to substitute the normal point correspondence requirement for camera pose estimation. Only four small movements are needed to extract the camera-tracker relationship. Later in [MH03], the authors used pure translational motion for both the camera intrinsic parameters and Hand-Eye transformation recovery. Bianchi et al. [BWH05] incorporated both visible and infrared (IR) LEDs to eliminate the measurement between the world and tracking frame. Andreff et al. [AHE99] used a structure from motion algorithm to retrieve the camera pose. By doing so, the Hand-Eye calibration method becomes online. Dornaida and Horaud [DH98] recovered the Hand-Eye and the world-base transformation in a unified formulation.

Starting with the next chapter, we present our contributions to the field of AVR, addressing video completion as our first topic.

### 2.4 HDR imaging

Research in High Dynamic Range (HDR) Imaging remains active in the computer graphics and computer vision communities. Most previous research has focused on the generation and representation of HDR images given multiple snapshots taken under identical scene-camera geometric relations. In order to generate a HDR image, the camera response function, which defines the mapping from the irradiance to the image brightness, needs to be recovered.Debevec and Malik [DM97] construct the camera response function while the reciprocity holds (as long as the production remains the same, halving the irradiance and doubling the exposure time will not change
the optical density). Their approach requires a minimum of two images to recover the camera response function, though more images are needed to generate a whole range radiance map. Khurram Shafique and Mubarak Shah [SS] model the camera response function for each color channel as a gamma curve. This approach requires a set of registered images under different unknown illumination conditions. A rough estimation of exposure settings for a set of registered images is also enough to extract the camera response function, as is proposed in Mitsunaga and Nayar’s work [MN99].

Currently, generic display devices have a relatively low dynamic range compared with the radiance range in a natural scene. In order to display the HDR radiance map plausibly, researchers have proposed different tone mapping approaches ranging from linear mapping to non-linear, perceptibly meaningful techniques. One typical tone mapping approach was presented by Reinhard et al. [RSS02].

The basic requirement of multiple registered images for HDR imaging limits the practicality of HDR techniques in video representation, due to the high possibility of a moving camera. Global motion compensation was the first response from researchers to extend HDR imaging into video [MN99]. Kang et al. [KUW03] combine an affine global model with optical-flow local adjustment to register differently exposed frames. The problem of possible occlusion between the consecutive frames is not addressed.

The color inconsistency between differently exposed frames needs to be addressed before applying most vision-based correspondence searching algorithms. The color transfer approach
[RAG01a] is popular. Porikli [Por03] calibrates the inter-camera colors based on a color correlation map. Though promising, this approach helps very little in over-saturated and under-illuminated regions, because the detail information in those regions is "flattened" by the Low Dynamic Range camera.

Hongcheng Wang et al. [WRA05] make a detour from image registration by customizing a camera rig with three CCD sensors aligned on the same principal axis. Their results achieve good spatial and temporal consistency, though the camera rig is not readily available.
CHAPTER 3
VIDEO COMPLETION

The ability to remove large objects in videos is critical to many applications, such as video editing and post-production. Given an input video, the goal is to remove the undesired objects and reconstruct the corresponding unknown regions in the entire video sequence based on motion information. However, most current approaches [DCY03b, BS00, BVS03, CPT03, JT03, LZW03, IP97, She03] focus only on region completion in a single image. In this work, we present a novel approach that is applicable to video sequences containing several planar motion layers. Our method is based on the assumptions that the overlapping order of the motion layers in each frame is maintained throughout the input videos, and that there is no cross occlusion between the layers in the video. For example, given that a video that contains three layers 1, 2, and 3, if 1 occludes 2, and 2 occludes 3, 3 cannot occlude 1.

Based on this assumption, we first apply a level-set representation and graph cut approach to achieve motion layer extraction. By exploiting the occlusion order constraints on multiple consecutive frames, the occluded pixels and the layer ordering are also explicitly determined. We then remove the undesired layer (the large object) and locate the corresponding unknown areas in other layers for every frame. After selecting the reference frame, we apply motion compensation to partially or even fully fill the unknown region in each layer. For the layers where some regions are
still missing, we develop a graph cut based region completion algorithm to complete the missing data with the perceptually correct color-texture information. Finally, based on the layer motion parameters, we project the synthesized layers to render each new frame. Figure 3.1 illustrates our algorithm including the intermediate steps.

This chapter is organized as follows. Section 3.1 addresses the details of our video completion algorithm. In Section 3.2, we demonstrate three sets of results obtained by our approach. 3.3 presents a summary of the results demonstrated in this chapter.

3.1 Object Removal in Videos

Given an input video sequence, our goal is to remove relatively large objects in the video and fill in the removed area with reasonable color-texture information in all of the frames. Our algorithm consists of three main steps: (1) The video sequence is segmented into different motion layers, and the overlapping order among the layers is then determined. (2) After removing the undesired object (one of the layers) in the video, the missing region in each layer of the reference frame is completed by motion compensation and region completion. (3) The completed layers in the reference frame are warped into every video frame to fill in the missing region of the frames.
Figure 3.1: (a) One frame of the original video. (b) The result of the motion layer extraction. The layers are numbered based on their overlapping order. (c) and (e) All the layers except the one we want to remove. (d) and (f) Layers synthesized by applying motion information of all the frames in the video. (g) Pixels which are still unknown are filled with our region completion method. Bottom: Selected frames from the original and the synthesized video.
3.1.1 Motion Layer Extraction

Given an input video, our segmentation algorithm can determine the number of the motion layer, and extract the layers accurately based on the motion parameters (affine or projective transformation).

In our approach, we first detect the seed correspondences over a short video clip [XS03]. Each patch around a seed correspondence is considered as an initial planar layer in the scene. The region’s boundary is then gradually propagated along the normal direction using a bi-partitioning graph cut algorithm integrated with the level-set representation. Using this approach, we can effectively filter out the bad seed regions. At this stage we apply a two-step algorithm to merge these initial layers into several groups, such that each group shares a single motion transformation. However, this layer merging method may not provide a correct segmentation of the scene, and the non-textured areas may belong to several layers due to their ambiguities, such as seen in the white paper at the lower part of the calendar in the mobile-calendar sequence shown in Figure 3.6.
A graph cut algorithm is applied with an occlusion order constraint on multiple consecutive frames to obtain accurate layer segmentation [XS04]. At the same time, we also explicitly determine the occluded pixels. In our graph cut framework, this multi-frame motion segmentation problem is formulated as an energy minimization problem of the following function,

\[
E = \sum_{j=1}^{n-1} (E_{\text{smooth}}(f) + E_{\text{data}}(f) + E_{\text{occ}}(f)) + \sum_{j=1}^{n-2} E_{\text{order}}(f),
\]

where \( j \) is the frame number, and \( n \) is the total number of frames (\( n \) usually is set to \( 3 - 5 \)). In this equation, \( E_{\text{smooth}} \) and \( E_{\text{data}} \) are standard terms in graph cut algorithms [BVZ01, KZ02], which correspond to a piecewise smoothness penalty and data error penalty, respectively. The other two terms are related to occlusion energy. The first one is \( E_{\text{occ}}(f) \), which is used to impose the occlusion penalties for the occluded pixels between frames 1 and \((j + 1)\). The second one is \( E_{\text{order}}(f) \), which is used to impose occlusion order penalties for maintaining the occlusion order constraint on each consecutive pair of images. After applying the graph cut algorithm, we extract the precise motion layers and explicitly identify the occlusion pixels between the overlapping layers as shown in Figure 3.2.

Given the motion layer segmentation and the occlusion information between each pair of layers, we use a simple approach to extract the overlapping order among the layers. For every pair of overlapping layers \( \Gamma_p \) and \( \Gamma_q \) in frame \( F_i \), we denote the overlapped area by \( \rho_{pq} \). If the correspondent area of \( \rho_{pq} \) in frame \( F_{i-1} \) belongs to \( \Gamma_p \), \( \Gamma_p \) is on top of \( \Gamma_q \) or vice versa. Based on this scheme, every layer in the video is assigned an order number, where the background layer is
always assigned the number 1. Figure 3.3 demonstrates the order of the layers in a mobile-calendar sequence. Since the ball layer and the train layer do not overlap, and are on top of all other layers, they are numbered as a single layer.

3.1.2 Layer Compensation and Completion

Since each motion layer extracted from the previous step contains either a distinct object or the whole background, it is easy to remove the undesired object by deleting the corresponding layer. After removing the undesired layer, \( i \), all the layers with smaller order numbers may have missing regions in some frames of the video. For each uncompleted layer, a motion model is applied to find the motion parameters between frames. Then, in each layer, a compensated reference frame is generated by warping all of the frames together with their motion parameters respectively as shown in Figure 3.1.\( d \) and \( f \). In most cases, there may still be some large portions of a layer missing color-texture information. In order to fill in the remaining missing regions, we propose a graph cut based single image completion method.

Before we explain the proposed method in detail, a few terms need to be defined. A known area is a region with all color-texture information available. An unknown area is a region with no color-texture information. A source patch is a small neighborhood area fully contained in a known area. A target patch has the same size as the source patch, and is located on the boundary of an unknown area. We denote the known area by \( \Phi \), the unknown area by \( \Omega \) and the boundary of \( \Omega \) by \( \partial \Omega \). The source and target patches are denoted by \( \Psi_s \) and \( \Psi_t \) respectively.
Our method is based on non-parametric texture synthesis. Therefore the filling order of the patches is critical to the quality of completion. In order to keep the performance at a reasonable level, we randomly select one patch from among a few potential patch locations on $\partial \Omega$ containing the largest known region, and define it as the target patch, $\Psi_t$.

After determining the target patch, $\Psi_t$, a patch matching step is applied to find a source patch, $\Psi_s$ in $\Phi$, which has the best similarity with $\Psi_t$. We define the center of the previous target patch, the current target patch, the previous source patch and the current source patch as $x_{t1}$, $x_{t2}$, $x_{s1}$ and $x_{s2}$, respectively. If $x_{t1}$ and $x_{t2}$ are close enough, $x_{s2}$ has a very high possibility to appear around $x_{s1}$ due to the spatial similarity assumption. Therefore we can reduce the search space $\Phi_s$ from $\Phi$ to a neighborhood area around $x_{s1}$, if the distance between $x_{t1}$ and $x_{t2}$ is within a threshold.
The similarity between the two patches can be expressed directly as follows:

$$
\Psi_s = \arg \min_{\Psi_i \in \Phi_s} \frac{d(\Psi_t, \Psi_i)}{N_t},
$$

(3.2)

where the distance $d(\Psi_t, \Psi_i)$ between the two patches is defined as the sum of squared difference (SSD), and $N_t$, the number of pixels in the known area of the target patch, $\Psi_t$, serves as a normalization factor.

This simple approach works well for images without projective deformation. However, for natural images, this condition may not be effective. In order to handle the deformation, we estimate the projective transformation parameters between the two patches based on [MP95] before applying the similarity measurement.

After estimating the projective transformation parameters, the patch $\Psi_s$ is warped to $\Psi_s'$ based on the motion parameters. Therefore the similarity measure can be calculated between $\Psi_t$ and $\Psi_s$. We then propose a new framework to update the patch $\Psi_t$ using $\Psi_s'$. Instead of formulating this problem as a merging problem, as done by previous researchers [CPT03, JT03, DCY03b], we reformulate it as a cutting problem: given two similar and spatially overlapping patches, where should a cut be made to separate those two patches and to make the seam least noticeable?

In our case, the patches can be cut only in the overlapping region, $\Psi_o$, where both patches have known information. We define each location in the overlapping region as a vertex $v_i$. Let $C_t(v_i)$ and $C_s(v_i)$ be the color value at the location $v_i$ in $\Psi_t$ and $\Psi_s$ respectively. The bi-partitioning problem can be solved by minimizing the energy

$$
E = E_{\text{smooth}}(f) + E_{\text{data}}(f)
= \sum_{(v_i, v_j) \in \mathcal{N}} W(v_i, v_j) + \sum_{v_i \in \Psi_o} D_p(f_i),
$$

(3.3)
where $D_p$ is set to constant and the weight function, $W(v_i, v_j)$, between vertices $v_i$ and $v_j$ is defined as follows:

$$W(v_i, v_j) = \begin{cases} 
\|C_t(v_i) - C_s(v_i)\| + \|C_t(v_j) - C_s(v_j)\| & \text{if } \{v_i, v_j\} \in \mathcal{N}, \\
\infty & \text{otherwise}
\end{cases}$$

(3.4)

where function $\| \cdot \|$ denotes the Euclidean distance between color values, and $\mathcal{N}$ is a 4-connected neighborhood. After defining the weight function as above, the minimal cut can be easily computed by a standard graph cut algorithm. A small weight means that if the cut runs between the pair of vertices, the four resulting color pairs $C_t(v_i)$ and $C_s(v_j)$, $C_s(v_i)$ and $C_t(v_j)$, $C_t(v_i)$ and $C_t(v_j)$, and $C_s(v_i)$ and $C_s(v_j)$ do not have much difference. Therefore, the cut gives the least noticeable seam. On the contrary, a large weight between two vertices implies that a seam between the two vertices is more noticeable. Figure 3.4 shows two results for our region completion approach.

### 3.1.3 Frame Composition

After the layer compensation and completion, all of the color-texture information for each synthesized layer is available in the reference frame. The next step is to project the synthesized layers to render each new frame based on the layer motion parameters, which can effectively maintain the temporal consistency for all the frames in the video.

For each frame, $F_i$, in the video, the motion parameters with respect to the reference image can be computed by accumulating motion parameters of consecutive frames. Therefore, the pro-
3.2 Experiments

We tested our approach on three video sequences, mobile-calendar, car-map and statue-road, to demonstrate the effectiveness of the method. Figure 3.6 shows the results of five selected frames of the well-known mobile-calendar sequence. Layer motion parameters are computed by using affine models. Each layer has its own distinctive motion (calendar moving down, background moving...
Figure 3.5: Left Top: A target frame in which a large region of the car layer is covered by the map board. Right Top: The reference frame in which the car layer is fully visible. Left Bottom: Synthesized car layer by warping the car images based on the directly calculated motion parameters. Right Bottom: Final result of our method by applying the constant velocity constraint.

right and ball and train moving left). Our results fully demonstrate the advantage of incorporating motion layer segmentation into the video completion framework.

Figure 3.7 shows the results selected from a car-map sequence. This sequence challenges our approach in several aspects: (1) The background has a strong perspective projection deformation. (2) The map board not only occludes the background but also occludes the moving car. A large portion of the body of the car is covered by the board in a few frames in the sequence. In this case, directly computing the motion parameter gives noticeable misalignment after warping, since the car layer in different frames has different available areas. The left bottom image in Figure 3.5 shows the results. In order to refine the result, a motion prediction approach is applied. We
use a constant velocity assumption and apply it as the initial condition for the motion estimation. By doing so, a much better result is achieved and shown in the right bottom image in Figure 3.5. Besides the compensation, our method successfully recovers the background and the full car body in every frame of the sequence. We did not remove the shadow of the map board because it does not interrupt the perceptual appearance of the output video. It can be easily removed by our method if desired.

The last results in Figure 3.8 are selected from a statue sequence. The original sequence is taken by a hand-held camera. The motion between frames is not smooth. Our method can still remove the statue in the scene in an unnoticeable manner.

3.3 Conclusions

A novel method is presented in this chapter to solve the problem of object removal in videos. Our contributions mainly focus on three areas: (1) Incorporating the motion layer segmentation method into our framework. This not only segments the motion layers, but also retrieves the overlapping order among the layers. This is very crucial for correctly rendering the synthesized layers in the missing regions. (2) Introducing graph cut in single image completion to improve the quality of the completion results. (3) Applying layer motion compensation to maintain the completion consistency in the video sequences.
Figure 3.6: Top row: Five selected frames from the original mobile-calendar sequence. Bottom row: The correspondent frames obtained by our algorithm, in which the train and the ball are removed.

Figure 3.7: Top row: Five selected frames from the original car-board sequence. Bottom row: The corresponding frames obtained by our algorithm, in which the map board is removed.

Figure 3.8: Top row: Five selected frames from the original statue sequence. Bottom row: The corresponding frames obtained by our algorithm, in which the statue is removed.
4.1 Introduction

The original blue screen keying technique can be traced back to the 1950s when the movie industry started to develop methods to extract foreground shots over a blue background, then composited a new background with the foreground. This was a time-consuming approach that includes filtering the blue background for a negative matte, then inverting the matte to acquire a foreground matte. The ideal goal for a blue screen technique is to composite a configurable new background into the scene, such as from a separate live camera input, a recorded video playback, or a digital offline or online source. The lack of camera pose knowledge limits the early stage of blue screen to only fixed cameras.

In the digital video age, the background color generally changed from blue to green. One of the reasons for this change is that the green color in digital format normally can produce higher luminance than the blue color. For many cases, the color that is chosen for the background is based on the requirements for the foreground colors. For example, for a general outdoor scene with no people in the scene, a red color can be a good choice even though it produces a poor matte for human skin. For simplicity and clarity, in this chapter we still use blue screen keying/matte as the formal name of this research direction, even though it now represents a broader concept.
Recently, developments in vision-based camera pose tracking have relaxed the constraint for a fixed camera pose for blue screen keying. Camera-visible markers are placed into the configured scene. In post-production, shots taken by calibrated cameras can be tracked and the relative camera pose to the referred objects can be discovered. In a high-cost blue screen setup, the ideal conditions can be met with uniform lighting and background color, a high quality camera and a carefully selected set of foreground colors. Under these conditions a commercial-level key matte can be generated, even in realtime. This realtime capability makes the blue screen technique extendable for interactive applications.

One of the major tasks that a MR system needs to accomplish is the seamless mixing of the real scene with the virtual content. When the application has knowledge of what content in the real scene needs to be blended into the mixed view, a blue screen technique can effectively pick up the unwanted parts in the scene by coloring them into a predefined uniform key color [SHM02]. However, typical MR applications normally involve a head-mounted display (HMD) with low camera quality, and a large range of material being colored by the key definition. This in turn requires the keying algorithm to be robust in the presence of high noise levels and variations of the color appearance. As mentioned in chapter 2, existing blue screen matte algorithms consider only the range of color for key generation. The spatial knowledge that the background color is designed to be uniform is not considered. Based on an assumption that the color variation at the foreground-background boundary area is noticeably higher than the background color variation, we consider image gradient to be an important clue on blue screen keying and explicitly define the
tri-map (the foreground region, background region and an unknown region) to reduce the influence of isolated noise.

We propose a three-stage keying algorithm to reduce the noise in the key output. First, a standard blue screen keying algorithm is applied to the input to get a noisy key, and the corresponding key region boundary is extracted. Second, the image edge information is calculated and the edge map is compared with the boundary result from the first step to define a confidence map. Finally, a tri-map with dynamic unknown band width is constructed based on the confidence map. A gradient based matting approach is then applied on the unknown region to complete the key generation.

This method works quite well when the foreground objects define a solid boundary and the keying algorithm can generate a key region close enough to the edge boundary. Isolated noise can be effectively removed as demonstrated later in the results section. However, a noisy blue-spilled foreground sometimes causes the key algorithm to generate regions with boundaries far from the edge. This in turn breaks up the continuity of the confidence map and results in a failed tri-map construction. In order to overcome this deficiency, we refine our approach by focusing not only on the gradient edge information, but also on the uniform color region assumption for the background. This extension stabilizes the tri-map construction, while effectively reducing the keying error caused by camera noise and incorrect keying algorithm classification.

In our new algorithm, a fast non-isotropic interpolation is applied to the original image to smooth out the random camera noise without affecting the edge contrast. A relaxed keying algorithm is then applied to the smooth image to pick up the absolute background region. This step
does not require the generated key to match the real edge boundary of the original image. Instead it requires only that the key is guaranteed to be in the background, a condition that can be achieved by selecting only the absolute “blue” color as the background. After this step, a hybrid region growing algorithm seeded by the key matte is applied to acquire the background region. This step alternatively selects the edge information and a combination of edge information and local color consistency as the constraint to the growing scheme, which in turn reduces the influence of spike color noise that can exist in a very small region. Figure 4.1 illustrates both algorithms that we proposed in this chapter. All added and modified steps in the second algorithm are drawn in red with the original approach is drawn in black.

The proposed approach in this chapter introduces several contributions. (1) Instead of using only the color range information as the traditional blue screen algorithms do, we first generate a tri-map based on a color consistency assumption and apply a natural image matting algorithm on the tri-map. This two-step method effectively stabilizes the keying output with respect to the camera noise. (2) We introduce the region growing concept into the blue screen keying approach, which can easily incorporate high-level image information into the evaluation, thereby giving it an advantage in the presence of background variation compared with the traditional blue screen algorithms. (3) We improve the seeded region growing algorithm (SRG) by alternatively selecting the edge constraint and the color variation constraint based on the growing shape condition. This in turn reduces the influence of the isolated spike color caused by camera noise.
The contents in the remainder of this chapter are organized as follows. Section 4.2 describes the core algorithm for the edge only keying algorithm. We then show keying results in the cases where the confidence edge map is closed and where it is not. In section 4.3 we propose the extended algorithm combining both edge and color variation information to generate a key matte. A few sets of results are shown in section 4.4, ranging from synthetic images to real video taken by our MR head mounted display camera. Conclusions are drawn in section 4.5 including current limitations of our algorithm. Potential future work is also discussed in this section.

4.2 Edge Based Tri-Map Construction and Keying

4.2.1 Chroma Keying Algorithm

The requirement for this step is to generate a key output that can closely reflect the foreground boundary. Noise is acceptable and will be handled by later steps. We propose a PCA-based chroma keying algorithm that can stabilize the key output with respect to the background color spectrum. We classify the background color space using geometric objects whose boundaries limit the opacity range. Classifiers are shaped into simple objects as ellipsoids to achieve the required computational efficiency, and are fit to the background training color cloud. We precondition the color space by using PCA, resulting in a new coordinate system such that the center of the key color spectrum is the origin and each axis is maximally de-correlated with respect to the optimal background color axis decision. In the training stage, we collect multiple (typically two to four) images containing a subset of the background color spectrum. A PCA operation is then applied to construct a Euclidean
Figure 4.1: Scheme overview of the algorithms we proposed in this chapter. The extended parts are colored in red.
transformation for the color cloud. In this new color space, the probability of a color being within the key color spectrum decreases radially with respect to the distance from the origin. In this isotropic color space, the optimal inner decision boundary, containing colors within the background color spectrum, and the optimal outer decision boundary, constraining the key color spectrum, can be defined by simple parameterized shapes. To directly exploit the isotropic nature of the trained space, we define two spheres parameterized by their radii. In an $RGB$ color space, these are represented by ellipsoids centered at the mean of the training data. When the user adjusts the radius of a sphere, the amount of color within the key color spectrum changes proportionally to the principal colors present in the training data.

Given the decision boundaries determined in the training session, computing the alpha key is implemented in realtime using a pixel shader. For a given sample point, its opacity is based on its relative distances between the boundaries of the inner and outer regions. By clamping this value to the range $[0,1]$, any point within the inner region is treated as transparent ($\alpha=0$); any outside the outer region is opaque ($\alpha=1$); and any other point has opacity based on the percentage of its distance from the inner to outer region boundary. We use the Euclidean distance from the origin with linear interpolation between the two decision boundaries to determine the opacity value,

$$\alpha = \frac{||s|| - r_{in}}{r_{out} - r_{in}},$$

(4.1)
Figure 4.2: (a) A noisy input image taken by our head mounted display camera. (b) Key result by our method. (c) Composition result with a white background.

where $r_{in}$ and $r_{out}$ are the radii of the inner and outer ellipsoids. One typical frame of the result by our method is shown in Figure 4.2. Isolated noise still exists in the key result, especially in the lower-left corner.

A detailed discussion about this part of the work can be found in [BZC07]. In practice, a Euclidean distance measurement approximation can be calculated to classify the background color [BL99] using

$$2B - R - G > \alpha$$  \hspace{1cm} (4.2)

where $R,G$ and $B$ represent the red, green, blue channel for the input color respectively.

4.2.2 Generating Key Boundary Map with Confidence and Constructing Tri-Map

Uniform Boundary In order to construct a tri-map for the matting step, a set of closed boundaries on the keying result needs to be extracted. We apply a morphological operation in the gen-
erated key region. Given a binary background key $K$, a uniform key boundary map $B$ can be generated by

$$B = \delta(K - K \otimes h, \sigma)$$ (4.3)

where $h$ is an averaging filter kernel that propagates the key region uniformly; and $\delta(a, b)$ is a binary function that returns 1 if $a > b$; otherwise it returns 0.

**Edge Map**  
Edges on the input image carry gradient information that explicitly silhouette the boundary of foreground objects and their internal texture variations. Therefore they serve as strong cues to separate the foreground from the consistent background. A straightforward way to extract edge information is to threshold the magnitude of the image gradient

$$g(x, y) = \sqrt{I_x^2(x, y) + I_y^2(x, y)}$$ (4.4)

This strategy ignores the continuity of edges and produces thick edges, which are not desirable for our purposes. In order to detect both strong and weak edges, and get less influence from the noisy input, we select a Canny operator for edge detection. A Canny edge operator is a multi-stage calculation that mainly consists of the following steps:

- Because the edge detection algorithm is a gradient-based approach, it is sensitive to image noise. Therefore before applying actual edge calculation, a smooth filter needs to be convolved with the image to reduce noise. A typical Gaussian filter is selected for this purpose.
• After the noise reduction step, the magnitude of the gradient is approximated by a Sobel operator, in which the absolute gradient magnitude along vertical and horizontal directions are added to simulate the gradient magnitude. The edge direction is then calculated and discretized into four directions.

• An operation called *non-maximum suppression* is applied to the edge content to eliminate each weak edge point that is not the maximum along the gradient direction.

• Instead of using only one threshold, a pair of thresholds $T_1$ and $T_2$ are selected so that $T_1 > T_2$. Any edge point that has its intensity value larger than $T_1$ is labelled as an edge first. Then each edge point that is connected with the labelled points along gradient directions is also labelled as an edge if its intensity value is larger than the smaller $T_2$. This heuristic operation effectively recovers the weak edges connecting with the strong ones.

We denote the calculated edge map as $E$. The edge map closely represents the boundary of the foreground objects. Ideally, boundary map $B$ should overlap $E$ on the boundary. However a few factors cause the calculated key boundary to be distant from the edge map. These are noise in the input, background color spill to the foreground objects, and background color variation. Therefore the distance between the points on the key boundary and their neighbor edges gives a confidence value for the boundary point that measures how accurate the point is on the actual foreground boundary. We calculate the confidence values by

$$C = K \cdot (E \otimes g)$$  \hspace{1cm} (4.5)
where $g$ is a Gaussian kernel convolving an edge map. We acquire the confidence boundary as the core of the overlapping region between the foreground and background. The confidence map is then propagated into its neighborhood area uniformly. As a consequence, a lower confidence boundary point grows wider than a higher confidence point does. In this manner, a dynamic unknown band is generated. Since the boundary map is closed, a hole filling algorithm [Soi99] can complete the foreground region from the boundary. At this point, a tri-map with a dynamic width unknown band has been constructed.

**Matting in the unknown region**  After a tri-map is constructed, the key values in the foreground region and the background region are defined as 1 and 0, respectively. Any existing blue screen keying technique can be applied to the unknown region for keying. However, all of these suffer the same problem as in the first step due to camera noise. For this reason, we choose a gradient-based matting algorithm, hypothesizing that these are more reliable for noisy images than color-based methods.

Recall that the regular blue screen compositing equation is:

$$C = \alpha F + (1 - \alpha)B$$  \hspace{1cm} (4.6)

where the pure foreground color and background color are linearly interpolated with one key factor $\alpha$. The derivative of $C$ with respect to $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ is

$$C' = \alpha'(F - B) + \alpha F' + (1 - \alpha)B'$$ \hspace{1cm} (4.7)
Within the area around the foreground boundary, the key value $\alpha$ changes faster than both the foreground color $F$ and background color $B$ do. In this case, equation 4.7 can be rewritten as

$$C' \approx \alpha'(F - B)$$

and therefore,

$$\alpha' \approx \frac{C'}{F - B} \tag{4.9}$$

When the difference between the foreground and the background can be approximated as $D_a$ (subtracting the average neighbor color in the foreground from the color in the background meets this requirement), equation 4.9 becomes a guidance vector field $\alpha'$. Key value $\alpha$ in the unknown region $\Phi$ can be solved by minimizing the following formula:

$$\min_{\alpha} \int \int_{V \in \Phi} ||\alpha' - \frac{C'}{D_a}||^2 d\mathbf{v} \tag{4.10}$$

The Dirichlet boundary condition requirement for solving this problem can be easily defined since on the background boundary $\alpha = 0$, and on the foreground boundary $\alpha = 1$. A Gauss-Seidel iteration method can be applied to this problem to get a unique solution. A detailed discussion for the Poisson Matting approach can be found in [SJT04].

### 4.2.3 Algorithm Evaluation

We test the proposed approach with video sequences taken by the same head-mounted display camera but with different background colors, blue and green. Figure 4.3 shows one of the typical frames for the blue screen sequence. Our algorithm successfully removes most of the noise existing
in the regular color-based keying algorithm. It is clear to see from Figure 4.3 (b)-(e) that the image edge map gives enough confidence to the keying-out boundary that a closed silhouette for the person is maintained. From the tri-map image in Figure 4.3 (f) we can find an unknown region located at the top-left corner, resulting from a relatively large noise region, which produces an edge boundary in the edge map. The gradient-based Poisson Matting approach assigns a very small key value to this region since the color variation is minimum, which contributes limited foreground data into the composition image in Figure 4.3 (h). However, as we mentioned earlier in this chapter, this algorithm relies on a close relationship between the key boundary and the image edge. This cannot always be achieved as we show in Figure 4.4. In this experiment, the person in the scene wears light-colored trousers that reflect a noticeable amount of the green color from the background. As a consequence, the edge detection algorithm does not determine the edge correctly on the foreground boundary. Although from the boundary confidence map (e) we can clearly see that a large misclassified background region is correctly labelled with 0 confidence, the real foreground boundary is also broken due to an insufficient edge map. This broken boundary cannot be used to construct a tri-map; therefore the algorithm fails as shown in the keying output in (f). We propose in section 4.3 an extended version of our algorithm that can relax the requirement for a precise edge map and near perfect keying input.
Figure 4.3: (a) The original input image. (b) Key mask result based on our described PCA-based algorithm. (c) Canny Edge detection result. (d) Boundary map based on the key output. (e) Boundary map with the edge agreement confidence. (f) Tri-Map generated by the confidence boundary map. (g) Compositing result based on the PCA key over a green background. (h) Compositing result of our algorithm, note that the noise in (g) are all successfully removed. (i) One more compositing result over the background in our lab.
Figure 4.4: (a) The original input image. (b) Key mask result based on the plane cut algorithm. A large portion of the background is classified wrong. (c) Canny Edge detection result. Due to the similarity of the foreground color in the trouser's area to the background color, the detected edges are not located closely on the actual foreground boundary. (d) Boundary map based on the key output. (e) Boundary map with the edge agreement confidence. Part of the boundary has 0 confidence. That causes the boundary to be broken into non-connected pieces. (f) Key output by the boundary confidence map. Since the boundary is not closed, the morphological image filling algorithm cannot be used to construct a correct tri-map.
4.3 Refined Blue Screen Keying

4.3.1 Anisotropic Image Filtering

A large part of the image noise is due to the camera signal noise. A simple smooth filter like a Gaussian filter uniformly blurs all content in the image in both the uniform color region and the edge boundary. For our task, a filter that can smooth the image content without affecting the strong edges is preferable since the background has a constant color while the foreground is assumed to be distinguishable in the boundary area. A bilateral filter [TM98] meets the requirement nicely. It is an anisotropic filtering process to smooth images except on the strong edges. The basic idea about a bilateral filter is that it combines both color range and spatial distance information together to assign weight for filtering convolution. That is to say, in image $I$, for a given image location $p$, a location $q$ gets its weight by the Euclidean distance between $p$ and $q$ and the similarity of the color $I(p)$ and $I(q)$. To combine both spatial and color range distance, the filter is designed as

$$I_b(p) = \frac{1}{N_b} \int \int I(q) d(p, q) r(I(p), I(q)) dq$$  \hspace{1cm} (4.11)

with a normalization term $N_b$ as

$$N_b = \int \int d(p, q) r(I(p), I(q)) dq$$  \hspace{1cm} (4.12)

where $d(\cdot)$ is a distance function between $p$ and $q$ and $r(\cdot)$ represents a color similarity function. For a simple case, both functions can be designed as Gaussian.

Although a bilateral filter is a potential candidate and is not an iterative approach as other anisotropic filtering techniques are, its application is still limited by its computational complexity.
Figure 4.5: Two filtered images of the input from Figure 4.3 and Figure 4.4 by the fast bilateral filtering algorithm. Most of the camera noise is reduced or removed perceptually.

The fact is, the convolution kernel needs to be calculated for every pixel, in comparison to the one time calculation for the kernel of the Gaussian filter. In order to overcome this limitation, we employ the approach from [PD06] that converts the computation into a spatial-range multi-dimensional linear convolution. Using this technique, the image spatial directions $x -$ and $y -$ are joined with the intensity channel and a fixed multi-dimension filter kernel can be applied to it. This is a close approximation for the original bilateral filter that theoretically speeds up calculation about 16 times, and can be implemented on the GPU, thereby supporting our realtime goal. Two filtered results are shown in Figure 4.5. Perceptually, the noise level in the images is low and the edge boundary is properly maintained.
4.3.2 Hybrid Seeded Region Growing

For a typical blue screen keying algorithm, the main goal is to discover the relationship between the given pixel color, the estimated foreground color and the background color. No high-level image information is employed. This scheme works fine for well-defined high quality blue screen photos, but is not good enough to handle noisy data. In fact, a blue screen setup assumes the background region is theoretically uniform and the foreground is separable from the background. This gives a clue as to why the edge information needs to be included to effectively distinguish the two regions, since a uniform region as the background ideally produces no edge for edge detection. Inspired by [FMS04], we design a hybrid Seeded Region Growing (SRG) algorithm that effectively applies edge information together with color constraint to detect the background region. As explained in the literature, SRG techniques contain two critical aspects: how to select the seeds to grow and how to decide the region to assign.

In a blue screen setup, an advantage we have is that the theoretical background color is predefined, even though the actual camera acquisition may be varied. Since our goal for this step is to define the background region, correct background color selection is the only thing we need. This can be easily achieved by aggressively setting the parameters of a blue screen algorithm to only cut out a background region with extreme confidence. Figure 4.6 shows the seeded region achieved by the plan-cut blue screen keying algorithm with a rigid constraint to guarantee the output is correctly in the background. Even in this extreme case, our growing algorithm correctly recovers the background.

53
Figure 4.6: Left: The initial seed region for Figure 4.3 acquired by calculating $2B - G - R > 0.6$. This constraint is so strong that only a few pixels remain in the background region as seeds. The color is inverted for better visualization. Right: With the limited seeds, our SGR algorithm correctly grows the background into the proper area.

Given the set of background seeds $S_b$, the background region is started as $B_b = S_b$. An iterative seed growing approach is applied that, at each step, allows one pixel to be grown into the background region when it meets the merge condition. Initially, each pixel that is not in the seed region is labelled as an undefined pixel and each in the seed region is labelled as a defined pixel. We define the member of set $d\Omega$ as each undefined pixels that is adjacent to at least one defined pixel:

$$d\Omega = \left\{(x, y) \in \bigcup (B_b \cap N(x, y) \neq \emptyset)\right\}$$  \hspace{1cm} (4.13)

where $N(x, y)$ is the immediate neighbor area around $(x, y)$. For each $(x, y) \in d\Omega$, we define the local color variation merge condition as

$$C(x, y) = \left\| \bar{T}(\Pi_u(x, y)) - \bar{T}(\Pi_d(x, y)) \right\|$$  \hspace{1cm} (4.14)
where \( \Pi_u \) is a neighborhood region around \((x, y)\) including \((x, y)\) in the undefined region; \( \Pi_d \) is a neighborhood region around \((x, y)\) in the defined region; and \( \bar{T}(\Pi) \) is the average color for a neighborhood \( \Pi \). The edge map is used to define the edge constraint as

\[
E(x, y) = \min_{(x_e, y_e)} d((x, y), (x_e, y_e) \in E)
\]

(4.15)

where \( d(\cdot) \) denotes the spatial distance between \((x, y)\) and point \((x_e, y_e)\) in the edge region \( E \). Both of the constraints contribute to the merge condition, which is set to meet the requirement of

\[
\lambda C(x, y) + \mu E(x, y) < \delta
\]

(4.16)

where \( \lambda \) and \( \mu \) are the importance factors for the color and the edge constraint. Equation 4.16 works well when the neighborhood areas \( \Pi_u \) and \( \Pi_d \) contain similar numbers of pixels. We call this a balanced situation. When balanced, average colors in both regions are similarly influenced by random noise. Therefore there is no bias for the color selection. In the cases where the two regions are unbalanced, e.g. the grown background region surrounds only a few pixels, a single noise can cause condition \( C(x, y) \) to be unreliable. We define a balance factor as

\[
f(x, y) = \frac{\min(\Pi_u, \Pi_d)}{\max(\Pi_u, \Pi_d)}
\]

(4.17)

and dynamically select the merge condition between equation 4.16 and equation 4.15 based on \( f(x, y) \). Figure 4.7 provides a simulated demonstration for both balanced and unbalanced cases.
Figure 4.7: A demonstration for balanced and unbalanced neighborhood area. In the balanced case, both $\Pi_u$ and $\Pi_d$ are noisy but the average colors represent the potential growing possibility. In the unbalanced case, only one noisy pixel has its color away from the background region. However, it contributes a lot for the average color in $\Pi_u$ that may stop the growing in this case.

In Figure 4.8 we show that, with only color or edge merge conditions, the growing result cannot be optimum in the boundary region. The hybrid scheme effectively takes advantage of both conditions. The balance factor dynamically selects the merge conditions in the case where isolated colors cause problems for the combined merge conditions, and the result is clearly displayed in Figure 4.8.

A similar step is applied to generate a tri-map in this case as we did in section 4.2.

### 4.4 Results and Comparisons

As a first step to evaluate the performance of our algorithm, we construct a synthesized image with a pure blue background and two distinctive foreground color regions. In this example we know the ground truth for both the foreground region $F$ and the background region $B$. We apply
Figure 4.8: One frame result to compare the growing conditions. (a) SRG by only the color constraint without balance factor. Note that the background region grows into the trouser’s area since the neighborhood color in that area is similar. Without the balance factor, many isolated regions cannot be correctly grown into. (b) SRG by only edge constraint. As the edge result in (d) shows, a small portion of the left bottom in the foreground is not connected well. Therefore this edge only result grows the background into the foreground area. (c) SRG by hybrid growing. The boundary of the foreground area is properly maintained.

A zero-mean Gaussian noise with a variation $\sigma$ up to 0.12. When $\sigma = 0.12$, the foreground region contains strong noise that is similar to blue color, and the background noise change the the original blue color to all over the RGB color space. We calculate the keying value and fairly classify the calculated foreground region $F_c$ when $\alpha > 0.5$; otherwise the location belongs to the calculated background region $B_c$. The accuracy rate of our foreground calculation is defined by

$$r_f = \frac{\Psi(F \cdot F_c)}{\Psi(F)}$$

where $\Psi(\cdot)$ is an area function calculating the size of the region, and operator “$\cdot$” is conceptually similar to the dot product, which maintains the region that is correct in $F$. Similarly, a high accuracy rate for the background calculation can be achieved. In our experiment, the foreground accuracy is 1 (perfect) and the background accuracy is around 0.983 (over 98% accurate). This provides strong evidence that our algorithm works in the presence of a high noise level.
Figure 4.10 and Figure 4.11 each show six key frames of the compositing results based on our keying algorithm. The output has a small amount of noise while maintaining the foreground boundary.

Figure 4.12 shows one frame that challenges previous keying algorithms. The photo is taken by the Canon VR2002 head mounted display camera with an initial resolution at $640 \times 480$. The lighting condition for the setup is poor, causing the dark shadow area to merge into noise. This setup makes it extremely hard to separate the regions. The frame on top is the original image. We also reduce the size of the input to $320 \times 240$ for comparison purposes. Five algorithms run on this input and the key mattes are shown in Figure 4.12. Visually it is easy to see that our proposed approach generates the least noise in the output, though it misclassifies a very small portion of the image. Given our goal to optimally reduce noise in the keying result, the proposed method meets our criterion very well. It is interesting to observe that the lower resolution input produces results with smoother boundaries. Our observation is that this may be caused by the isotropic filter applied to the image to reduce size, which in turn also reduces the noise.

### 4.5 Conclusion and Future Work

In this chapter we illustrate our efforts to generate a proper blue screen key under noisy conditions. The goal is to smooth out the keying noise without losing the correct foreground region. We first propose a three-stage keying algorithm to reduce the noise in the key output based on gradient edge information. This method effectively removes most of the color noise when the initial keying
output matches the edge information. However, a closed region boundary may lose its continuity when the color match is weak. In order to overcome this deficiency, we apply a seeded region growing algorithm that obeys constraints imposed by background region color consistency and acquired edge information. The consequence is that the quality of the initial keying result can be heavily relaxed so long as the selected regions are guaranteed to be in the background. The new algorithm performs well even when the edge map has significant discontinuities.

Though promising, the seeded region growing algorithm requires iteration that may lead to hundreds of repetitions before converging on a final solution. The potential for a significant number of iterations can make the quantity of texture look ups prohibitively high. This limits the performance of the proposed keying algorithm. In order to speed up the process, the iteration time can be bounded based on the GPU’s computational power. In this case, the final growing stage can take a fixed number of iterations, which in turn can make it possible to achieve interactive performance.
Figure 4.9: (a) A synthesis image with pure blue background. Both the background and foreground regions are known as ground truth. (b) Image result by adding zero mean Gaussian noise with variation 0.12 to (a). (c) Plane cut result for $2G - B - R > 1.9$. By this high threshold the foreground region is free of wrong classifications. (d) Keying result by our algorithm. (e) It shows that when the noise variation increases, our algorithm achieves accuracy rates in the foreground and background close to 1 and remains constant.
Figure 4.10: A compositing sequence with blue background. The image on the top is one of the original frames, while the other six images are selected compositing frames.
Figure 4.11: A compositing sequence with green background. The image on the top is one of the original frames, while the other six images are selected compositing frames.
Figure 4.12: This figure shows the results of one of the hardest blue screen inputs. The original frame shown in the top are taken in a low light condition, therefore noise and shadow merge together and it is very hard to distinguish them even interactively. From left to right, top to bottom: our result with the input resolution at $320 \times 240$; our result with the input resolution at $640 \times 480$; our real time result described in section 4.2.1; result by plane cutting algorithm from section 4.2.1; result based on Vahols’ patent; result by a commercial keying software called combustion.
CHAPTER 5
CAMERA-TRACKER CALIBRATION

Accurately deriving the transformation between a head-mounted display and an affixed 6-DOF tracker is critical for mixed reality applications. This transformation brings the HMD cameras into the tracking coordinate frame, which in turn overlaps with a virtual coordinate frame to create a plausible mixed visual experience. In this chapter, we present a novel camera-tracker calibration method by reformulating the classic Hand-Eye calibration problem into a camera pose estimation with just one known 3D reference point. Two benefits can be achieved by this reformulation. First, a 3D-2D correspondence mapping is used to evaluate the minimization error. This fits the requirements of MR applications, in which the quality of the merging of the virtual and the real scene is the goal, even if the estimated 3D pose may contain larger errors. Second, for the camera pose estimation problem, a well-defined robust estimation can be applied to reduce the influence of tracker reading errors.

5.1 Introduction

Most Mixed Reality applications combine a head-mounted display (HMD) with at least one type of tracking system, which provides a 3D pose of the camera with respect to a predefined tracking coordinate frame. An MR system can then integrate the pose of the real camera to merge virtual
contents correctly onto the real video stream. In order to stabilize the virtual content and minimize the alignment error between the virtual and real input, the quality of the tracking system needs to be high, and the transformation between the camera and the tracker needs to be precisely recovered.

Here, we focus on accurately recovering the transformation between the camera and the affixed tracker for the purpose of MR applications. Many researchers classify the problem as a Hand-Eye Calibration [TL88b, PM94, BWH05], a term that originated within the robotics community, where a camera as the “eye” is mounted on a robot “hand.” Given a minimum of three stable Hand-Eye poses, the camera-tracker transformation can be expressed by an equation $AX = XB$, where $X$ is a homogenous 4 matrix representing the unknown transformation, $A$ describes the change of the poses for the camera, and $B$ is the corresponding change for the robot arm.

For a MR application, the 2D-2D virtual-real alignment is critical to the mixing quality. Even though the estimated 3D pose of the camera has a relatively large error, as long as the 2D alignment is correct, the user will not notice the disparity. Keeping this in mind, we formulate the camera-tracking problem as a 3D camera pose estimation problem inspired by [TGN02], which requires only one known land marker in the tracking coordinate frame and minimizes the 2D projection error between the image observations for the land marker in different camera poses. We use a minimized parameter set to express rotations, therefore simplify the minimization formula. Also, this formulation enables us to apply a robust estimation for the minimization procedure to reduce the influence of suspicious image observations, thereby resulting in an estimated transformation that is more robust in the presence of measurement error.
This chapter is organized as the follows. Section 5.2 explains a straightforward but not very accurate calibration approach. Section 5.3 explains the formulation to convert the camera-tracker calibration into a 3D camera pose estimation. Section 5.4 describes the two-step pose estimation approach, which starts with a simple linear estimation that is then refined by a non-linear robust estimation with a Tukey estimator. In Section 5.5 we show the estimation results and compare the work with other Hand-Eye methods. Section 5.6 concludes the work and points out future research directions.

### 5.2 Naive Camera-Tracker Calibration

As [BJB03b] noted, the transformation between the tracker to the camera, $M_{tc}$, is hard to measure due the fact that the tracker center and camera center are not physically marked. Fortunately, the transformation of the tracking system to the world, $M_{Tw}$, tracker to tracking system, $M_{tT}$, and world to camera, $M_{wc}$, can be reasonably recovered. Our approach is based on the observation that $M_{tc} = M_{tT}M_{Tw}M_{wc}$. Figure 5.1 demonstrates the coordinate frame transformation in our MR system.

By carefully measuring the location of a few points in both the tracking system frame and the world (model) frame, the transformation between the world frame and the tracking system frame $M_{wT}$ can be linearly solved. The tracker to tracking system matrix is a direct read from the tracker. Although $M_{wc}$ cannot be directly measured manually, we align a calibration grid pattern in the world coordinate frame in a manner so that one corner of the grid pattern is on the origin.
Figure 5.1: An illustration for the possible coordinate frames existing in MR system. From the figure, it is straightforward that the transformation between physical tracker(sensor) and cameras can be recovered when the world model can be measured in tracking frame. In reality, measuring a position (LED light point) in tracking frame is much easier than measuring a full transformation between world and tracking frames.
of the world frame, and two adjacent edges on the grid pattern are parallel to the world frame axes $X$ and $Y$, respectively. Therefore, the extrinsic parameter recovered from the calibration can be transferred directly as $M_{wc}$. However, the results generated from this method is depended on correctly measuring a set of parameters in the world frame with respect to the tracking frame, which normally gives noisy result.

### 5.3 Camera Pose Formulation

Though the measurement between the world frame and the tracking frame can be hard to acquire since the orientation alignment over the position measurement requires careful user interactions, precisely locating one land marker (in our case, a LED light) is achievable. Figure 5.1 shows the relationship graph between the relevant coordinate frames. For our one point calibration method, we assume that one land marker can be precisely measured in the tracking frame; therefore we only need three 3D coordinate frames: tracking system (T), tracker(t), and camera(c). One 2D coordinate frame for the image is defined as (i). The letter in the subscript of an entity gives the coordinate frame in which the entity is represented. For example, $X_t$ represents a 3D point in the tracker frame, and $R_{tc}$ is the rotation matrix from the tracker to the camera.

For a fixed homogenous point $X_T$ in the tracking system frame, its image observation can be calculated from it by a $3 \times 4$ projective matrix. However, for different camera poses, the corresponding projective matrices are not the same. Since the tracker is solidly attached to the
camera, 3D points in the tracker frame and their 2D observations can be mapped by

\[ x_i \cong K[R_{tc}|t_{tc}]X_t \]  \hspace{1cm} (5.1)

where \( K \) is the camera intrinsic parameter matrix and \( R_{tc} \) and \( t_{tc} \) represent the rotation and translation between the tracker and the camera. By observing equation 5.1, it is clear that the 3D points in the tracker frame do not have to be read in the same camera pose. Every 4 × 4 tracker pose matrix \( M_{Tt} \) transforms a point \( X_T \) in the tracking system frame into the tracker frame as \( X_t \). Since all movements are relative, we can consider a movement of the tracker with respect to the tracking system frame as a movement of the tracking system frame with respect to the tracker. Therefore one known point in the tracking system frame is good enough to form a point cloud with \( n \) points in the tracker frame as long as we move the camera-tracker rig accordingly.

For a camera with known intrinsic parameters \( K \), we now have \( n \) correspondences between known 3D points and their 2D image observations. This converts the camera-tracker calibration problem into a 3D camera pose estimation problem.

### 5.4 Two Steps Pose Estimation

The 3 × 4 projective matrix \( P \) between \( X_i^t \) and \( x_i^t \) can be estimated by a linear system called a Direct Linear Transformation (DLT). Since \( x_i^t \cong PX_i^t \), vectors \( x_i^t \) and \( PX_i^t \) have the same direction and the cross product between the two is zero. With some reorganization, the equation can be written in the form of
\[
\begin{bmatrix}
0^\top & -x^t_{i3}X_t & x^t_{i2}X_t \\
x^t_{i3}X_t & 0^\top & -x^t_{i1}X_t \\
-x^t_{i2}X_t & x^t_{i1}X_t & 0^\top
\end{bmatrix}
\begin{bmatrix}
P^\top_1 \\
P^\top_2 \\
P^\top_3
\end{bmatrix}
= 0 \tag{5.2}
\]

where \( P_i \) represents the \( i \)-th row of \( P \). Since the three equations are linearly dependent, we can omit the third row in the equations and turn it into a form of

\[
A_1P = 0 \tag{5.3}
\]

where \( A_1 \) is a \( 2 \times 12 \) matrix. A minimum of 6 3D-2D independent correspondences are needed to solve the equation uniquely. For \( n \) pairs of correspondences, we stack up the \( 2n \) equations to obtain a \( 2n \times 12 \) matrix \( A \). The solution is the eigenvector of \( A^\top A \) with least eigenvalue. A normalization constraint \( \|(p_{31}, p_{32}, p_{33})^\top\| = 1 \) is applied for minimizing the algebraic error. Once \( P \) is available, the transformation between tracker and camera can be extracted up to a scale factor,

\[
[R_{tc}|t] \sim K^{-1}P. \tag{5.4}
\]

Since the rotation matrix \( R_{tc} \) is not guaranteed to be orthogonal, a correction can be made as mentioned in [Zha00]. Basically, a corrected rotation matrix can be calculated by

\[
\hat{R}_{tc} = UV^\top \tag{5.5}
\]

where \( U S V^\top \) is the singular value decomposition of \( R_{tc} \).

Tuceryan et. al. [TGN02] applied ideas that are similar to our linear method when addressing optical see-through Hand-Eye calibration. However, their approach requires more extensive user interaction.
Due to the noisy measurement on both the 3D and 2D points, the DLT algorithm can not get a useful result directly. Therefore we take the result from the DLT algorithm as an initial estimation for the pose, and apply a non-linear robust estimation approach to refine the algorithm.

The 6 DOF redundancy in the rotation matrix representation makes using $R_{tc}$ directly for non-linear estimation a clumsy choice. To overcome this problem, we select an exponential map representation for the rotation, recognizing that a gimbal lock can be easily avoided in this representation and that this approach reduces the constraint complexity in comparison to approaches using a unit quaternion representation.

An exponential map converts a 3-vector $\vec{v} = [v_1, v_2, v_3]^\top$ and its magnitude $||\vec{v}||$ to a corresponding rotation. Giving the skew-symmetric matrix $\Omega$,

$$
\Omega = \begin{bmatrix}
0 & -v_3 & v_2 \\
v_3 & 0 & -v_1 \\
-v_2 & v_1 & 0
\end{bmatrix}
$$

(5.6)

the rotation matrix $R_{tc}$ can be expressed as

$$
R_{tc}(\Omega) = I + \frac{\sin(||\vec{v}||)}{||\vec{v}||} \Omega + \frac{(1 - \cos(||\vec{v}||))}{||\vec{v}||^2} \Omega^2
$$

(5.7)

To simplify the description, we write $R(\vec{v}) = R_{tc}(\Omega)$. For the noisy measurement, we refine the camera pose estimation by minimizing the sum of the reprojection error, which is the distance between 3D-2D projections of the land marker and the image observations. We write
Figure 5.2: A closeup view of our camera-tracker rig for the MR system and the testing configuration for our one point algorithm. The left image shows the LED light solidly mounted on a calibration checkerboard. The checkerboard is used for Park-Martin’s method. The right image show the actual camera-tracker rig setup. The tracker’s coordinate frame is demonstrated in red color and the camera frame is in blue.

\[
(\vec{v}, t) = \arg \min_{(\vec{v}, t)} \sum_i \text{dist}^2(K[R(\vec{v})|t|X_i^1, X_i^1]) \tag{5.8}
\]

It is well known that a least-square minimization is very sensitive to gross error due to the fact that it does not differentiate outliers. We add a robust estimator called a Tukey estimator to our minimization, which is defined as

\[
\rho_t(r) = \begin{cases} 
\frac{c^2}{6} \left[ 1 - \left(1 - \left(\frac{r}{c}\right)^2\right)^3 \right] & \text{if } |r| \leq c \\
\frac{c^2}{6} & \text{otherwise.}
\end{cases}
\tag{5.9}
\]

Compared to other M-estimators, a Tukey estimator might not reliably converge to the global minimum; however, it can discard outliers that have large residual errors. This is a preferred
approach when the tracker quality is limited or the image observation is noisy. With the Tukey estimator, equation 5.8 can be rewritten as

$$\begin{equation}
(\tilde{v}, t) = \arg \min_{(\tilde{v}, t)} \sum_i w_i^2 \text{dist}^2(K[R(\tilde{v})|t]|X_i^1, x_i^1)
\end{equation}$$

(5.10)

where $w_i$ is a weight calculated in every iteration by

$$w_i = \frac{\rho_i(\text{dist}(K[R(\tilde{v})|t]|X_i^1, x_i^1))^{\frac{1}{2}}}{\text{dist}(K[R(\tilde{v})|t]|X_i^1, x_i^1)}$$

(5.11)

The minimization is achieved by applying Levenberg-Marquardt algorithm, the details of which are described in Appendix A.

### 5.5 Results and Comparison

The real images used in our experiments are taken by camera with intrinsic parameters calibrated, and the radial and tangential distortion rectified in advance to validate the pin hole camera assumption. The offline camera calibration has been thoroughly researched [CF98, Zha00, HCD00]. The details for calibration are not in the scope for this chapter. The basic idea for calibration is to capture images of a planar guide board with a known grid-shaped pattern. With a correct estimation of correspondences between the observation and the 3D grid model, the intrinsic parameter matrix $K$ can be calculated by iterative least-squares estimation. A camera calibration toolbox is available online at

www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html, which can generate optimized in-
trinisc parameters and rectify the distorted images. Figure 5.3 illustrates the accuracy of the calibration for the extrinsic parameters. The checkerboard shown in the figure is used only for comparison purposes.

5.5.1 Simulation

The parameters used in the simulation step are designed to reflect a realistic lab environment so we can simulate the actual physical setup. The unit for translation is set as a millimeter. We define a rotation matrix $R_{tc}$ as

$$R_{tc} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$ (5.12)

and a translation vector $T_{tc} = (30.0, -100.0, 0.0)^\top$. Accordingly, the exponential map vector for the rotation matrix is $(-1.2092, -1.2092, -1.2092)^\top$. This set of parameters closely mimics the real camera-tracker configuration shown in the right image of Figure 5.2. Based on this, a random set of 20 3D points in the tracker frame and their image projection in the image frame are generated. Both our DLT algorithm and the non-linear optimization approach can correctly calculate the result of the rotation and the translation. In the case of non-linear optimization, the algorithm is not sensitive to the initial estimation. For example, an initial exponential map vector $(-0.3, -0.3, -0.3)^\top$ and a translation $(10, -10, 20)^\top$ which is far away from the real transformation gives a converged
Figure 5.3: Top images shows the extrinsic calibration result for our camera. The camera poses shown in the image clearly demonstrates that the way we took images can cover a good range of space for our testing purpose. Bottom image gives the coordinate frame for the checkerboard.
result as

\[
[R_{tc}|T_{tc}] = \begin{bmatrix}
-0.0000 & 1.0000 & 0.0000 & 29.9826 \\
0.0000 & -0.0000 & 1.0000 & -99.9927 \\
1.0000 & 0.0000 & -0.0000 & -0.1493
\end{bmatrix}
\] (5.13)

In order to evaluate the robustness of our method with respect to the measurement noise, we adjust the simulation data set by adding a random offset on the 2D coordinates in image frame. The random offset is uniformly distributed in a window with an area of \(w \times w\) centered at the original 2D position. We use the average reprojection error as the comparison entry and compare the results of DLT only algorithm and our proposed method. Figure 5.4 shows the plot of the results with a window size from 0 to 30 pixels. It is clear to see that, with a non-linear optimization, the proposed method produces more stable results in comparison to a DLT only method similar to that in [TGN02].

A third simulation is designed to test the performance of the robust estimator. Two 2D measurement outliers are thrown into random positions of the 2D measurements in the original simulation data and the reprojection errors are compared between the results of the regular Levenberg-Marquardt method and the results produced with a Tukey estimator. We run this process for four times and the results are plotted in Figure 5.5. The robust estimator totally throws away the outliers and produces the correct results.
Figure 5.4: This plot shows the comparison between the average reprojection errors for the DLT only algorithm and the proposed approach. X-axis is the width of the offset windows $w$ and Y-axis is reprojection error.

### 5.5.2 Real Data Evaluation

In order to demonstrate the efficiency of our method, we also conduct a real calibration evaluation and compare our approach with a classic Hand-Eye method by Park and Martin [PM94], which we call the PM method. We solidly attached an Intersense IS900 PC tracker on top of our CANON VS2002 Stereo HMD. Figure 5.2 shows the configuration in our MR application system. For every fixed camera-tracker pose, a pair of images is captured. As Figure 5.6 shows, a dark image is acquired to extract the 2D LED location, and a bright image is taken to calibration the extrinsic
camera parameters for Hand-Eye calibration. A total of 40 images is recorded for 20 different camera-tracker poses. The average reprojection error of the PM method is 117.09 and the average reprojection error of our method is 69.40. A few typical results are shown in Figure 5.7.

5.6 Conclusion and Future Work

In this chapter, we present a novel camera-tracker calibration by formulating the problem into a camera pose estimation scenario. We represent the orientation with an exponential map, which minimizes the degree of freedom of the rotation parameter space. Therefore no orthogonal (for
rotation matrix) or unit (for unit quaternion) constraints are needed for non-linear minimization, which in turn simplifies the calculation. A non-linear robust estimation approach is applied to recover the transformation between the camera and the tracker. We demonstrate that the proposed approach gives promising results with respect to the 2D projection error. In the future, instead of assuming a known land marker in the tracking coordinate frame, we plan to calculate the camera-tracker calibration together with the land marker’s location in the tracking system frame. This should relieve the burden for the user to precisely measure a 3D location.
Figure 5.7: Three typical frames of the reprojection results. Blue dots in the images are our results, and the red dots are results by PM method.
CHAPTER 6
LUMINANCE RANGE ENHANCEMENT IN MIXED REALITY

6.1 Introduction

Current computer graphics rendering techniques are able to produce images that are close to photo-realistic. When these approaches are applied in an MR environment, it should be possible to make the virtual indistinguishable from the real. However, limited by the contrast ratio (around two orders of magnitude) of conventional output devices, the actual images displayed cannot match the range that occurs in many natural settings. For instance, an indoor scene that includes visibility to outdoor sunlight provides a contrast ratio of five orders of magnitude, a range that is within the capabilities of the human eye [RWP05] but beyond those of most conventional displays. To validate our luminance enhancement approach, one assumption must be made. When set at the same exposure level, the two cameras in the stereo rig present the same color for most scene objects. When the two sets of hardware for displaying the views of the left and right cameras are identical, the actual colors displayed on a camera are decided by illumination geometry and viewing geometry. Since the stereo rig takes images at the same moment, the illumination geometry for the scene is constant at that time. The distance between the two camera centers in our stereo rig is less than 70mm. For any scene position more than 1.5m from the stereo rig, the view angle difference $\alpha$ is less than 3 degree. In the well-known Phong lighting model, the specular reflection intensity
Figure 6.1: Layout of our Mixed Reality System – User acquires real scene from a video see-through HMD. Scene is processed using pre-scanned virtual geometry that, while not displayed, assists in the registration of the binocular images.

is proportional to the $n_s$ power of a cosine function of $\alpha$, which results in around 10 percent of intensity differences for 3 degrees even if $n_s$ is as high as 100. These facts render our assumption reasonable for a practical range of MR environments.

This chapter is organized as follows: Section 6.2 presents an overview of our framework, including the notations and the components of the system. Section 6.3 explains the construction and operations based on the concept of a Video-Driven Time-Stamped Ball Cloud (VDTSBC) model for registering stereo images. Section 6.4 shows how to generate an Enhanced Dynamic Range Video. Section 6.5 demonstrates and evaluates a few experimental video results. The final section of the chapter, 6.6, presents conclusions and future work.
6.2 Framework Overview

Instead of excluding all environmental elements as is done in virtual reality, MR treats the virtual and real environments as a cooperative pair. Although not exclusively a visual experience, vision, as a dominant sensory perception of human beings, plays a major role in MR.

For a MR experience to be delivered successfully, there are two types of camera models that need to be registered precisely. Figure 6.1 demonstrates the hardware setup for our system. We define R_cameras as the two real cameras mounted on the HMD, and V_cameras as the corresponding virtual cameras located in the virtual scene and superimposed on the R_cameras. The R_cameras are pre-calibrated in our system and the internal parameters are transferred to the V_cameras. The
external parameters, which define the camera orientation and position in the world coordinate frame, are acquired from a hybrid acoustical/inertial positional/pose tracker.

Figure 6.2 summarizes the pipeline of our video enhancement approach. The stereo camera rig calibration and the background point cloud scanning are applied offline. The outdoor region in the cloud is labeled based on the window areas. During an interactive experiment, the relative poses of the cameras to the background are updated by a permanently attached tracker. Upon registration, the VDSTBC provides geometrical information to generate a virtual disparity map. Differing from a real disparity map, the virtual one records not only the disparity values, but the corresponding ball labels (see Section 6.3, by which a depth ordering is established to segment the occlusion region). Finally all the regions are submitted to the luminance enhancement/adjustment module to deliver an improved scene rendering.

6.2.1 Notations

The camera model we selected in our system is a finite projective camera with radial distortion. The basic definitions are listed in Figure 6.3. A detailed description for the camera model is in 6.2.2. In our framework, a capitalized boldface letter $\mathbf{X}$ denotes a non-homogenous 3-vector in Euclidian 3D space. $\mathbf{X}$ represents the corresponding homogenous coordinates. Similarly, lowercased $\mathbf{x}$ and $\mathbf{x}$ denote the same concepts in Euclidian 2D space. A tilde letter pair $(\tilde{x}, \tilde{y})^\top$ denotes the normalized coordinates, and a letter with a subscript $d$ represents the distorted coordinates. The coordinates on 2D camera plane are denoted as $(x_p, y_p)^\top$, and the final pixel coordinates are denoted as $(x_i, y_i)^\top$. 
Figure 6.3: An overview of the camera coordinate frame – For a general projective camera model, the projection center is called camera center $C$. The plane passing $C$ and parallel to the image plane is the principal plane. The $Z$ axis for the camera coordinate system is defined as the principal axis. The point where the principal axis meets the image plane is called the principal point $p$. The actual image origin is defined as $O$.

For stereo matching and luminance enhancement, $I$ denotes an image and $I(x)$ is the color value at $x$. The corresponding position of $x$ in the second camera is denoted as $\hat{x}$. The radiance value of $I(x)$ is represented by $\hat{I}(x)$.

6.2.2 CAMERA MODEL

The camera model used in this chapter is a finite projective camera with radial and tangential distortion. Related definition is shown in Figure 6.3. A finite projective camera can be modeled by a $3 \times 4$ perspective projection matrix $P$ as

$$P = K[R|t]$$  \hspace{1cm} (6.1)
The $3 \times 3$ upper triangular matrix $K$ records the intrinsic parameters of the camera, which has a form of

$$K = \begin{bmatrix} f_x & s & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{bmatrix} \quad (6.2)$$

where $f_x = f m_x$ and $f_y = f m_y$ represent the focal length $f$ scaled by the number of pixels per unit distance $m_x$ and $m_y$ in the horizontal and vertical direction. $s$ is a skew parameter which normally is 0 as the $x$ axis and $y$ axis for the CCD camera are perpendicular. $(u_x, u_y)^\top$ represents the coordinates for the principal point $p$ in Figure 6.3.

The orthogonal rotation matrix $R$ and translation vector $t$ in equation 6.1 represent the camera coordinate frame in the world coordinate. Explicitly, $R$ denotes the orientation of the camera coordinate frame and $-R^{-1}t$ denotes the camera center in the world frame.

Having $P$, a 3D point $\vec{X}$ in world homogenous coordinate maps to the image homogenous frame at $\vec{x}$ by

$$\vec{x} = P \vec{X} \quad (6.3)$$

The above linear projective camera model only mimics a pin hole camera model. For real lenses straight lines in the world are not straight in the image anymore. In order to compensate this non-linear imaging effect of real lenses, two types of camera distortion models need to be considered: radial distortion and tangential distortion. for a 3D point $X_c = (X_c, Y_c, Z_c)^\top$ in the
camera frame, a normalized image projection

\[
\mathbf{x}_n = \begin{pmatrix}
\frac{X_c}{Z_c} \\
\frac{Y_c}{Z_c}
\end{pmatrix} = (x, y)^\top
\]  

(6.4)

Denoting \( r^2 = x^2 + y^2 \), radial distortion can be modeled by a 6-order polynomial

\[
L(r) = 1 + p_{r1} r^2 + p_{r2} r^4 + p_{r3} r^6
\]  

(6.5)

In the case that the camera lenses are not co-centered, a tangential distortion vector also needs to be calculated as

\[
\mathbf{x}_t = \begin{pmatrix}
2p_{t1}xy + p_{t2}(3x^2 + y^2) \\
p_{t1}(x^2 + 3y^2) + 2p_{t2}xy
\end{pmatrix}
\]  

(6.6)

The updated normalized coordinate

\[
\mathbf{x}_{n2} = D(x) = L(r)\mathbf{x}_n + \mathbf{x}_t
\]  

(6.7)

The updated normalized coordinate then will be used with intrinsic parameter matrix \( K \) together to get image coordinate. In the case to remove the distortion, ideal image coordinate with no distortion can be defined as \( \mathbf{x}_i' = (x_i', y_i', 1)^\top \), the corresponding real image coordinate \( \mathbf{x}_i = (x_i, y_i, 1)^\top \) can be acquired straightforward.
6.3 Left-Right Input Registration by VDTSBC Model

The main contribution of this work is the introduction of the VDTSBC to assist the construction of a high quality stereo matching, which is the central issue for video luminance enhancement. We define a ball $B$ as a 6-tuple:

$$B = < T, R, C_l, C_h, t, r >$$  \hspace{1cm} (6.8)

where $T$ denotes the three-dimensional coordinates for the center of the ball. $C_l$ and $C_h$ specify the color information for the ball, back-projected from two differently exposed stereo cameras. $t$ records the duration from the time $C_l$ and $C_h$ are projected to the present. $r$ is the radius of $B$, which is defined by a function $radius(t)$. $R$ records the camera’s direction at time $t$. A Video-Driven Time-Stamped Ball Cloud $G$ is defined as a set of balls:

$$G = \{ B_i = < T_i, R_i, C_{li}, C_{hi}, t_i, r_i > | i = 1..n \}$$  \hspace{1cm} (6.9)

For each ball $B_i$, the positional parameter $T_i$ is acquired by the offline geometry scan using a 3D laser scanner. Our approach to acquire $C_{li}$ and $C_{hi}$ is explained as follows.

To simplify the notation, we ignore the index $i$. For a ball $B$, the homogenous coordinates $\overrightarrow{X}$ in the world frame are given by $(T,1)^\top$. The homogenous coordinates $\overrightarrow{X_c}$ for $B$ in the camera
frame are presented by

\[
\overrightarrow{X}_c = \begin{bmatrix} R & -R\hat{C} \\ 0 & 1 \end{bmatrix} \overrightarrow{X}
\]  \hspace{1cm} (6.10)

where \( \hat{C} \) represents the coordinates of the camera center in the world coordinate frame. The ideal point corresponding to \( B \) in one camera view is presented by

\[
(\hat{x}, \hat{y})^T = (\overrightarrow{X}_c(1)/\overrightarrow{X}_c(3), \overrightarrow{X}_c(2)/\overrightarrow{X}_c(3))^T
\]  \hspace{1cm} (6.11)

In order to register the virtual scene correctly onto the camera view, a camera distortion model needs to be taken into consideration. For our non-wide angle stereo camera, it is not necessary to push the radial component of the distortion model beyond the 4th order. [Zha00] also suggests that the tangential distortion is negligible compared with the radial distortion; therefore the actual projected point of Ball \( B \) on the camera frame is calculated by \( (x_d, y_d)^T = L(r) \cdot (\hat{x}, \hat{y})^T \). Detailed explanation for \( L(r) \) can be found in 6.2.2. The coefficients \( a_1 \) and \( a_2 \) are the camera radial distortion parameters. The actual pixel coordinates of the Ball \( B \) are

\[
(x_p, y_p, 1)^T = K(x_d, y_d, 1)^T
\]  \hspace{1cm} (6.12)

where \( K \) is an upper triangular 3 \( \times \) 3 camera calibration matrix with the skew parameter preset to zero. The color information \( C \) is then acquired by a convolution around \( x_1 = (x_{p1}, y_{p1})^T \)

\[
C_k = G(\xi)I(x_1 + \xi) \hspace{1cm} k = l, h
\]  \hspace{1cm} (6.13)
where $G(\xi)$ is a spatial smoother, in our case a Gaussian convolution kernel with a small displacement vector $\xi$, and $I(x_i + \xi)$ denote the image color information at $x_i + \xi$. Under occlusion, a ball can be visible in one view but not in the other view. The $C_l$ and $C_h$ calculated in this case are not consistent using this approach. The reason is that, under occlusion, the projected color information to the ball $B$ belongs to some other balls that overlap $B$ from the camera view direction.

In order to handle occlusion, we extend the disparity map concept in the virtual scene to include one more parameter, the ball label $p$, which indexes the corresponding ball $B$ to generate the disparity. This extension builds a direct connection between a geometrical position and its image projections. On the other hand, the real disparity map preserves depth cues related to the cameras.

Keeping these ideas in mind for one camera, we project all balls in VDTSBC onto two cameras to generate disparity values. Without loss of generality, we only describe the virtual disparity map construction in the right $V_c$-camera. Assume ball $B$ has a projection $x$ in the right $V_c$-camera and a projection $\hat{x}$ in the left $V_c$-camera. The disparity is calculated by

$$\partial x = x - \hat{x}$$

(6.14)

If several balls have the same image projection $x$, the ball that has the smallest depth $\vec{X}_c(3)$ has its index attached with $x$, and its disparity value $\partial x$ is updated in the final virtual disparity map for the right $V_c$-camera. If the index for the corresponding position $\hat{x}$ in the right $V_c$-camera is not the same as for $x$, the index $p$ for $x$ is modified to a unique value that identifies the occlusion.


6.4 Enhanced Dynamic Range Video Generation

Theoretically, after registering most of the regions in the stereo images, a full radiance map that covers the whole image region is needed in order to enhance the image dynamic range. However, the possibility of missing registration cannot be ignored [KUW03]. In their case, Kang et al. relax the registration requirement by excluding over-saturated or under-illuminated pixels from the weighted radiance map generation function.

Our system distinguishes the indoor and outdoor environments by labeling the VDTSBC. Based on the labeling, we either enhance or adjust the luminance dynamic range to achieve the desired perceptual improvement.

For generating a correct tone map, existing standard methods like [RSS02] can be applied, given camera color calibration in advance. For the purpose of enhancing luminance range, our simple linear method works reasonably well and can be easily implemented in our online framework.

6.4.1 Luminance Enhancement

The techniques to enhance luminance dynamic range in the left and right cameras are symmetric. Without loss of generality, we explain only the right image luminance enhancement in this subsection. With the assistance of the VDTSBC, most of the modeled regions in \( I_R \) have three available colors from which to choose. For a position \( x \) in such region in \( I_R \), \( \hat{I}_R(x) \) is from the right image;
\( \hat{I}_L(\hat{x}) \) is from the corresponding location in the left image; and \( C_{hi} \) is from \( B_i \) in the VDTSBC. These three have weights of \( w_R \), \( w_L \), and \( w_B \), respectively. If the corresponding position \( \hat{x} \) of \( x \) is occluded in the left camera, \( \hat{I}_L(\hat{x}) \) is meaningless. The radiance map can be calculated as:

\[
R = \sum_{k=1}^{n} \bar{w}_k \hat{I}_k
\]  

(6.15)

where \( \bar{w}_k \) is the normalized weight and we abuse the notation a little bit to denote \( \hat{I}_k \) as one of the available radiance values. \( n \) is 3 when three colors are available; otherwise \( n = 2 \).

### 6.4.2 Luminance Adjustment

Under high brightness settings in our system, the outdoor environments are too bright to be registered based on our stereo method. At the same time, the VDTSBC gives no geometrical information for outdoor environments. These facts limit us to a practical solution. For the left image \( I_L \), which has a lower brightness setting, the color from \( \hat{I}_L \) in the outdoor region is used to fill the radiance map. For the right image, things are more complicated since it is necessary to get detailed information from the right image for an outdoor region.

Though it is hard to achieve a precise registration in an outdoor region, an approximation is acceptable for most situations. This is supported by the fact that a large portion of the outdoor scene is far enough away to be “indistinguishable” through HMD video cameras. In fact, with a image resolution of \( 320 \times 240 \), any geometrical locations 15 meters or more away from the stereo camera...
Figure 6.4: The sparse 3D point cloud is projected onto the left and right images. Each point in the cloud represents a cross on the checkerboard. A radial distortion model is applied to achieve accurate registration.

rig can result in merely sub-pixel displacement. In our current implementation, a direct copy from the right image to the left image in the outdoor region is applied, though a small displacement along the Epipolar Line for each pixel may be more precise. The actual tradeoff for the warping function compared with direct copying needs further evaluation.
6.5 Implementation and Results

We built our framework from the Mixed Reality System developed at the UCF Media Convergence Laboratory (Hughes et al., [HSH05]). The positional information for the VDTSBC that we apply in our MR system is either synthesized or acquired by a Riegl LMSF-Z420i 3D terrestrial laser scanner. The head mounted display we select is the Canon VH-2002 see-through model, on which an Intersense IS-900 ultra-sonic tracking sensor is permanently attached. The system runs on two Pentium 4 3GHz machines, one for capturing and tracking, the other for rendering. The video card we use is an nVidia 6800 with 256MB memory. For a VDTSBC model with 20,000 points, the system runs at 2 frames per second.

For all of the experimental sequences, the stereo rig was calibrated offline by a stereo calibration tool box available at http://www.vision.caltech.edu/bouguetj/calib_doc/. Only the 2nd and 4th order radial distortion parameters are considered, and the camera skew is set to zero. The displacement between the tracker center and the stereo rig is also compensated before registering the virtual and real cameras.

Since our method does not rely on high accuracy of image feature correspondence, the difference between the two brightness levels within the stereo cameras can be adjusted freely depending on the radiance range the user wants to cover. Figure 6.5 shows a typical brightness setting for our framework.
Calibration board test. In this sequence, a flat shaped ball cloud is applied. Figure 6.4 shows a sparse cloud model to demonstrate the accuracy of the cloud-image, image-image registration. Figure 6.8 is a representative frame from the test. Since our method takes radial distortion into consideration, the slightly distorted image pair is correctly registered and the enhanced left view (f) has a much better brightness level than the left input does. The resolution of the input in this test is 512x480.

Office Room MR environment sequence. The office room has an unobstructed window facing the sun during the day time. A single brightness setting is clearly not enough to cover most of the details inside and outside the room, as Figure 6.6 (h) and (i) demonstrate. The VDTSBC here serves as the intermediate 3D marker for stereo image registration, and provides proper color for the occlusion region. Since we assume that the lighting condition cannot change suddenly, the output enhanced image results are perceptually acceptable, including those in the occluding region. Figure 6.6 shows a few key frames of two office sequences taken at two different times.
of day. In this set of sequences, a non-flat model is used for the VDTSBC. The direct merged result in Figure 6.6 (f) clearly shows that the disparity on the front side of the box is noticeably larger than the disparity on the wall. For an unconstrained camera with a moving camera center, it is impossible to compensate for the disparity with a single global motion. Here the VDTSBC serves as an effective guide to generate a correct disparity map. For an indoor-outdoor mixed area in Figure 6.6 (h)(i), our result preserves the dark indoor details like the Chinese characters on the wall with the visible outdoor scene (a portion of a tree) in the same frame Figure 6.6 (g).

**Pinball game machine sequence.** In this sequence, instead of defining a global coordinate frame for the scene, we define the top left corner of the game machine as the local origin; the right-hand coordinate frame has x- and y- directions parallel to the sides of the machine. The transformation between the machine coordinates and the tracking system coordinates is adjusted interactively. The machine has a very complicated shaped surface. We scan the 3D shape of the pinball machine with the Riegl scanner; a view of the point cloud is shown in Figure 6.7. To demonstrate the efficacy of our method, we apply a relatively sparse model of the machine. The small black holes shown in the results are the consequence of low point density. Other than this, the enhanced frames show very good registration results (Figure 6.9).

### 6.6 Conclusions and Future Work

In this chapter, we present a framework to enhance the visible luminance range in a mixed reality environment. We introduce the Video-Driven Time-Stamped Ball Cloud (VDTSBC) into the
camera registration pipeline to assist the matching between stereo images and to provide a rational luminance enhancement in occlusion areas, which is hard to recover using a pure stereo-based algorithm.

In the future, we will combine the accuracy of the VDTSBC disparity and the flexibility of image-based disparity to eliminate the simplifying requirement of a stationary MR environment. In a realistic MR setting, multiple participants may share the MR system and portals in the background (windows and doors) may be opened unintentionally. Accordingly, the related image information cannot be directly mapped to VDTSBC since they are not coming from the same object any more. Therefore it is important to distinguish the modeled region (stationary region) and non-modeled region (moving or changed region).
Figure 6.6: In this sequence, a paper box is attached on the wall to demonstrate the advantage of our method. (a) Left input with the projected point cloud on it. (b) One frame of the enhanced result of our method. (c)-(f) Left: two frames of the our enhanced results. Right: the corresponding direct merged results. (g) A frame of the enhanced result with indoor and outdoor mixed view. (h) Left input. (i) Right input.
Figure 6.7: One view of the point cloud model for our pinball game machine. The three bright boards around machine are markers used for model registration.
Figure 6.8: (a) Left input. (b) Right input. (c) Disparity map generated by VDT SBC. Note that only the checker board area is modeled. (d) Directly merged result for the left display. The displacement is large due to the close camera position. (e) Merged left result of our method. The brighter checkerboard is achieved by combining the checkerboard from right image to the entire left image. (f) Result of the checker board only.
Figure 6.9: From left to right: three typical enhanced frames for the modeled game table on the top and the corresponding direct merged results in the bottom.
CHAPTER 7
CONCLUSION AND FUTURE WORK

In the previous chapters, we presented components of a framework that supports augmentation in visual reality, both in online and offline contexts. The first topic addressed is offline video completion or more specifically object removal and repair of frames in a video. Our novel contribution starts with incorporating motion layer analysis that isolates each segment and retrieves the overlapping order among the layers. This ordering is crucial to correctly rendering the synthesized layers in the missing regions. We then apply a graph cut algorithm to provide high-quality completion results for each frame. Finally, we apply motion compensation to achieve completion consistency in the video sequences. The consequence is an altered video sequence that exhibits a consistent, visually convincing result. The presented approach is limited by the assumption that the video input can be segmented by simple motion layers. This constraint can be relaxed if a rough geometry can be constructed in the un-completed region. However, that extension is not explicitly addressed in this thesis; it remains a topic for future consideration.

Our second area of concern is the problem of blue screen keying in online applications. Blue screen keying explicitly removes selected regions from video streams, replacing these with synthetic content. When the video stream comes from a head-mounted display, as part of a mixed reality setup, our goal is to combine virtual and real content, with the virtual content generally re-
placing a blue screen background or being inserted for blue screen masks in portals such as doors and windows or at specific places like on tabletops that are to display synthetic content. The problem we address is that keying in mixed reality occurs in the presence of noise, e.g., due to low-end cameras or poorly controlled lighting. This is very different from the situation commonly found in professional studios where lighting is controlled and video capture is done with high-end cameras.

To attack blue screen keying in mixed reality applications, we present a three stage keying algorithm that is based on gradient edge information. This method effectively removes most of the color noise when the initial keying output matches the edge information. However, a completed region boundary may be broken up when the match is not close enough. In order to overcome this deficiency, we apply a seeded region growing algorithm that obeys constraints imposed by background region color consistency and acquired edge information. The consequence is that the quality of the initial keying result can be relaxed as long as the selected regions are guaranteed to be in the background. The new algorithm performs well even when the edge map has significant discontinuities. The one downside with the growth algorithm is its dependence on many iterations before convergence. This can be offset by using fixed iterations to generate the eventual solution for online processing with the help of the rapid increase in computational power and the ease with which the algorithm can be implemented in the shaders of graphics processing units.

A third topic on which we focused is the task of accurately deriving the transformation between a head-mounted display and an affixed 6-DOF tracker. This transformation is critical for mixed reality applications, in which the HMD cameras must be brought into the tracking coordinate frame
that, in turn, overlaps with a virtual coordinate frame to create a plausible mixed visual experience. Our contributions include an extension of the single point calibration method, and the use of 3D-2D correspondence mapping to evaluate the minimization error. This approach addresses the primary requirements of MR applications, in which the quality of the merging of the virtual and real scene is the goal, even if the chosen technique results in a less accurate estimated 3D pose. Our method uses Direct Linear Transformation for an initial estimate and then applies a non-linear robust estimation in order to lessen the influence of potential measurement errors.

Though a simple process, measuring one point in a tracking frame may still introduce unnecessary errors. As part of our follow-on research agenda, we plan to extend the existing algorithm by considering the case where we do not know the single point’s position. For this case, the number of unknowns which need to be discovered are nine (three additional for the unknown marker location) rather than the six required in this thesis.

The last topic addressed in this dissertation is to enhance the luminance range limitations of typical MR HMD cameras to more closely match the range that occurs in natural settings. We apply stereo camera rig calibration and offline background point cloud scanning to ensure a high quality real-virtual registration. During an interactive experiment, the relative poses of the cameras to the background are updated by a permanently attached tracker. Upon registration, the Video-Driven Time-Stamped Ball Cloud model provides geometrical information to generate a virtual disparity map. Differing from a real disparity map, the virtual one records not only the disparity values, but the corresponding labels of “balls” within the cloud. Finally all the regions are submitted to
the luminance enhancement/adjustment module to deliver an improved scene rendering. A vision-based stereo algorithm can be used to extend the current algorithm to cover the areas that the point cloud cannot address.

Other than the modules that we discussed in this dissertation, one important fact still attracts our focus. There remains a great need to improve the registration of real-virtual 3D coordinate frames so they are precise and consistent. The quality of calibrated 3D trackers needs significant refinement. Vision-based camera pose estimation generally produces precise results, although it may fail and be hard to restart. A hybrid approach, combining vision-based estimation and the use of a 3D tracker, is a promising direction. In the future, we will explore the possibility of hybrid online camera pose recovery in a stereo head mounted display tracking system. The goal is to improve the pose estimation accuracy by dynamically altering the estimation between different cameras. The idea is motivated by the fact that MR applications often involve a stereo see-through HMD as the video capturing and display interface. The geometric relationship of the stereo video streams provides extra constraints for both natural feature tracking and pose estimation. In order to reliably restart a vision-based method, a collection of reference images can be taken to cover most of the viewing area. From these, a set of interesting points are extracted and their corresponding 3D world coordinates can be discovered. These 2D-3D correspondences serve as a guideline to restart the estimation if it fails and to reduce drifting of the estimation.

The 3D tracker data for every image frame can be recorded as the initial estimation for the camera pose. Then, interesting feature points can be extracted from both stereo camera frames
with only the features showing in both frames fulfilling the stereo constraints being recorded. The reprojection error between observed feature points and projected geometry features can be minimized in a robust least square fit scheme. In the case where the physical tracker fails or when the signal quality is low, a wide baseline match is applied to register the tracked features with the 3D world coordinates. The major difference between this potential direction and state-of-art approaches in the literature is that the second camera input supplies not only the stereo constraints for feature tracking, but reduces the pose estimation uncertainty found in the monocular vision-based methods by dynamically switching the cameras used for updating the pose between the left and right cameras. One of the most interesting extensions in this future direction is to keep multiple pose estimations for each incoming frame. This roughly defines the probability distribution function for the pose presentation, which can be important when the physical tracking system cannot return a uniform reading along all directions.

In general, while the research reported here has successfully addressed a number of challenging problems in augmented visual reality, there are still a large number of issues remaining, especially in the area of online augmentation. Some of these problems will be greatly lessened by advances in graphics and general purpose processors, but all require strong algorithmic and theoretical development.
APPENDIX

LEVENBERG-MARQUARDT NON-LINEAR OPTIMIZATION ALGORITHM
For a non-linear minimization problem with small or medium sizes of unknown, Leverberg-Marquardt Optimization algorithm converges faster than gradient descent or conjugate gradient methods, due to the fact that it is an semi-second order approach which approximates Hessian matrix by a Jacobian production, and smoothly transforms between gradient decent (first order approach) to Gauss-Newton approach (semi-second order).

Given a set of parameters $p$, a set of observation $b$, and a function $f$ to map $p$ to $b$, optimize $p$ so that the sum of the square distance

$$E = \sum_i ||f(p) - b||^2$$

becomes minimal.

As an iterative minimization procedure, Levenberg-Marquardt algorithm requires an initialization of $p$, denoted as $p_0$. For every iteration step, the estimation of $p$ is updated by

$$p_{i+1} = p_i + \Delta_i$$

where $\Delta_i$ is calculated heuristically to minimize $E$ at the iteration step $i + i$. For a regular Gauss-Newton algorithm,

$$\Delta_i = -(J^T J)^{-1} J^T (f(p_i) - b)$$

where $J$ is the Jacobian of $f$ at $p_i$. However, this quadratic rule assume linear updating which only is valid around the minimum. Therefore, Levenberg-Marquardt algorithm introduce a weight factor to alternatively select the update for $\Delta_i$ between the gradient descent and a Gauss-Newton method. The new calculation for $\Delta_i$ is
\[ \Delta_i = -(J^T J + \lambda I)^{-1} J^T (f(p_i) - b) \] (7.4)

where \( \lambda \) is a weight factor that when it is small, the algorithm performs like a Gauss-Newton approach. On the other hand, the algorithm turns acting like a gradient descent algorithm.

The Levenberg-Marquardt algorithm can be summarized as below:

1. Calculate \( \Delta_i \) by 7.4.
2. Calculate \( E \) based on \( p_i + \Delta_i \).
3. In case \( E \) decreases, the update is good and decrease \( \lambda \) by a significant factor.
4. In case \( E \) increases, the update causes problem and need to be recalculated. Increase \( \lambda \) by a significant factor and go to 1.
LIST OF REFERENCES


111


[Iki00] M. Ikits. “Coregistration of pose measurement devices using nonlinear total least squares parameter estimation.”, 2000. 18


J. Sun, Y. Li, S. B. Kang, and Shum H. “Flash Matting.” In ACM SIGGRAPH, 2006. 14


121