**Hasselt University**
transnationale Universiteit Limburg

# Real-Time View Interpolation for Eye Gaze Corrected Video Conferencing

Doctoral dissertation submitted to obtain the degree of
**Doctor of Science: Information Technology**
to be defended by

**Maarten Dumont**

on September 30, 2015

Promoter:
Prof. Dr Philippe Bekaert
Co-promoter:
Prof. Dr Ir Gauthier Lafruit

# Abstract

Conventional video conferencing (e.g. Skype with a webcam) suffers from some fundamental flaws that keep it from attaining a true sense of immersivity and copresence and thereby emulating a real face-to-face conversation. Not in the least does it not allow its users to look directly into each other's eyes. The webcam is usually set up next to the screen or at best integrated into the bezel. This forces the user to alternate his gaze between looking at the screen to observe his remote conferencing partner and looking into the webcam. It is this conflict between both viewing directions that stands in the way of experiencing true eye contact. This issue of missing eye contact is the central problem to solve in this dissertation.

**An Image-Based Approach** We opt for an image-based approach to solving our problem, meaning that we synthesize an eye gaze corrected image from real-world (live) captured images. In the newly reconstructed image, the user's gaze will be corrected and thus the conflict between viewing directions will no longer be present. By using live imagery, we avoid the more artificial look and feel of many previous solutions that employ model-based reconstructions or avatar-based representations. Specifically, we investigate three main view synthesis algorithms to reach our goal. This results in contributions to environment mapping, disparity estimation from rectified stereo, and plane sweeping.

By designing and implementing all algorithms for and on the GPU exclusively, we take advantage of its massive parallel processing capabilities and guarantee real-time performance and future-proof scalability. This strategy of exploiting the GPU for general – non-graphical – computations is known as general-purpose GPU (GPGPU) computing.

Although developed here to correct eye gaze in video conferencing, our algorithms are more generally applicable to any type of scene and usage scenario.

**Four Prototypes** We develop four different system prototypes, with each prototype relying on its specific (combination of) view synthesis algorithm(s) to reconstruct the eye gaze corrected image. Each view synthesis algorithm is enabled by a specific configuration of the capturing cameras, allowing us to arrange and present the prototypes according to increasing physical complexity of their camera setup.

**Maintaining Camera Calibration** Maintaining the calibration of those cameras, however, may pose a challenge for a prototype that can be subject to a lot of dynamic user activity. Therefore, we first develop an efficient algorithm to detect camera movement and to subsequently reintegrate a single displaced camera into an a priori calibrated network of cameras.

Assuming the intrinsic calibration of the displaced camera remains known (physical movement is reflected in the extrinsic parameters), we robustly recompute its extrinsic calibration as follows. First, we compute pairs of essential matrices between the displaced camera and its neighboring cameras using image point correspondences. This provides us with an estimate of a local coordinate frame for each camera pair, with each pair related to the real world coordinates up to a similarity transformation. From all these estimates, we deduce a (mean) rotation and (intersecting) translation in the common coordinate frame of the previously fully calibrated system.

Unlike other approaches, we do not explicitly reconstruct any 3D scene structure, but rely solely on image-space correspondences. We achieve a reprojection error of less than a pixel, comparable to state-of-the-art (de-)centralized network recalibration algorithms.

**Prototype 1: Environment Remapping** Our first prototype is immediately our most outside-of-the-box solution. It requires only the bare minimum of capturing cameras, namely a single one, together with a single projector for display. Drawing inspiration from the field of environment mapping, we capture omnidirectional video (in other words, the environment) by filming a spherical mirror (the northern hemisphere) and combine this – after a remap of the captured image – with projection on an identically-shaped spherical screen (the southern hemisphere). Both hemispheres are combined into a single full sphere, forming a single communication device that allows to capture from the top and display at the bottom.

The unconventional novelty lies in the observation that we do not perform image interpolation in the traditional sense, but rather compose an eye gaze corrected image by remapping the captured environment pixel-to-pixel. We develop the mathematical equations that govern this image transformation by mapping the captured input to the projected output, both interpreted as parallel rays of light under an affine camera model. The resulting equations are completely independent of the scene structure and do not require the recovery of the depth of scene. Consequently, they have to be precomputed only once, which allows for an extremely lightweight implementation that easily operates in real-time on any contemporary GPU and even CPU.

Unfolding the environmental reflection captured on a (relatively small) specular sphere yields omnidirectional imagery with a projection center located at the center of that sphere. Consequently, the user looks directly into the camera when looking at the center of the sphere and eye contact is inherently guaranteed. Moreover, the prototype effortlessly supports multiple users simultaneously, unveils their full spatial context and offers them an unprecedented freedom of movement. Its main drawback, however, is the image quality. It is severely diminished by limitations of the mathematical model and off-the-shelf hardware components.

**Edge-Sensitive Disparity Estimation with Iterative Refinement** Our second prototype, which we will present in a moment, relies heavily on our novel algorithm for accurate disparity estimation. We make three main contributions.

First, we present a matching cost aggregation method that uses two edge-sensitive shape-adaptive support windows per pixel neighborhood. The windows are defined such that they cover image patches of similar color; one window follows horizontal edges in the image, the other vertical edges. Together they form the final aggregation window shape that closely follows all object edges and thereby achieves increased disparity hypothesis confidence.

Second, we formalize an iterative process to further refine the estimated disparity map. It consists of four well-defined stages (cross-check, bitwise fast voting, invalid disparity handling, median filtering) and primarily relies on the same horizontal and vertical support windows. By assuming that color discontinuity boundaries in the image are also depth discontinuity boundaries in the scene, the refinement is able to efficiently detect and fill in occlusions. It only requires the input color images as prior knowledge, can be applied to any initially estimated disparity map and quickly converges to a final solution.

Third, next to improving the cost aggregation and disparity refinement, we introduce the idea of restricting the disparity search range itself. We observe that peaks in the disparity map's histogram indicate where objects are located in the scene, whereas noise with a high probability represents mismatches. We derive a two-pass hierarchical method, where, after analyzing the histogram at a reduced image resolution, all disparity hypotheses for which the histogram bin value does not reach a dynamically determined threshold (proportional to the image resolution or the histogram entropy) are excluded from the disparity search range at the full resolution. Constructing the low-resolution histogram is relatively cheap and in turn the potential to simultaneously increase the matching quality and decrease the processing complexity (of any local stereo matching algorithm) becomes very high.

Implementation is done in CUDA, a modern GPU programming paradigm that exposes the hardware as a massive pool of directly operable parallel threads and that maps very well to scanline-rectified pixel-wise algorithms. On contemporary hardware, we reach real-time performance of about 12 FPS for the standard resolution ($450 \times 375$) of the Middlebury dataset.

Our algorithm is easy to understand and implement and generates smooth disparity maps with sharp object edges and little to no artifacts. It is very competitive with the current state-of-the-art of real-time local stereo matching algorithms.

**Prototype 2: Stereo Interpolation** Our second prototype turns to rectified stereo interpolation. We mount two cameras around the screen, one to the left and one to the right, and let the user be seated in the horizontal middle. We then interpolate the intermediate (and thus eye gaze corrected) viewpoint by following (and extending) the depth-image-based rendering (DIBR) pipeline. This pipeline essentially consists of a disparity estimation and view synthesis stage. The view synthesis is straightforward and very lightweight, but relies heavily on accurate disparity estimation to correctly warp the input pixels to the intermediate viewpoint.

On the one hand, the prototype is able to synthesize an eye gaze corrected image that contains very sharp and clearly discernible eyes. On the other hand, its reliance on stereo matching also gives rise to its biggest disadvantages. First, the user is restricted to move on the horizontal baseline between the left and right cameras, which causes eye contact to be difficult to maintain. Second, the small baseline preference of dense stereo matching forces us to either place the cameras around a smaller screen or assume a larger user-to-screen distance to avoid too large occlusions.

**Prototype 3: Plane Sweeping** Our third prototype aims to overcome these shortcomings by mounting six cameras closely around the screen on a custom-made lightweight metal frame. The more general camera configuration avoids large occlusions, but, as such a configuration is no longer suitable for rectified stereo, we must turn to plane sweeping to interpolate the eye gaze corrected image. The flexible plane sweeping algorithm allows us to reconstruct any freely selectable viewpoint, without the need of image extrapolation. Combined with a concurrently running eye tracker to determine the user's viewpoint, this ensures that eye contact is maintained at all times and from any position and angle.

A number of carefully considered design and implementation choices ensures over real-time performance of about 40 FPS for the SVGA resolution ($800 \times 600$) without noticeable loss of visual quality, even on low-end hardware.

First, from our strategy for disparity range restriction, we devise a method to efficiently keep a uniform distribution of planes focused around a single dominant object-of-interest (e.g. the user's head and torso) as it moves through the scene. A Gaussian fit on the histogram of the depth map will indicate the depth (mean) and extent (standard deviation) of the object. We can use this to retroactively respond to movements of the object by dynamically shifting a condensed set of planes back and forth, instead of sweeping the entire space with a sparser distribution. This not only leverages the algorithmic performance, but also implicitly increases the accuracy of the plane sweep by significantly reducing the chance at mismatches.

Second, we present an iterative spatial filter that removes photometric artifacts from the interpolated image. It does so by detecting and correcting geometric outliers in the jointly linked depth map that is assumed to be locally linear.

Third, we use OpenGL and Cg to reprogram the GPU vertex and fragment processing stages of the traditional graphics rendering pipeline, which better suits the inherent structure and scattered memory access patterns of plane sweeping. We even further improve the end-to-end performance by developing granular optimization schemes that map well to the polygon-based processing of the traditional graphics pipeline.

Finally, a fine-tuned set of user-independent parameters grants the system a general applicability. The result is a fully functional prototype for close-up one-to-one eye gaze corrected video conferencing that has a minimal amount of constraints, is intuitive to use and is very convincing as a proof-of-concept.

**Prototype 4: Immersive Collaboration Environment** Our fourth and final prototype is realized after recognizing that current tools for computer-supported cooperative work (CSCW) suffer from two major deficiencies. First, they do not allow to observe the body language, facial expressions and spatial context of the (remote) collaborators. Second, they miss the ability to naturally and synchronously manipulate objects in a shared environment.

We solve these issues by integrating our plane sweeping algorithm for eye gaze correction into an immersive environment that supports collaboration at a distance. In doing so, we identify and implement five fundamental technical requirements of the ultimate collaborative environment, namely dynamic image-based modeling, subsequent reconstruction and correction for rendering, a spatially immersive display, cooperative surface computing, and aural communication.

We also propose our last adaptation of the plane sweeping algorithm to efficiently interpolate a complex scene that contains multiple dominant depths, e.g. when multiple users are present in the environment. This time, we interpret the cumulative histogram of the depth map as a probability density function that describes the likelihood that a plane should be positioned at a particular depth in the scene. The result is a non-uniform plane distribution that responds to a redistribution of any and all content in the scene.

Our final prototype truly brings together many key research areas that have been the focus of our institute as a whole over the past years: view interpolation for free viewpoint video, calibration of camera networks, tracking, omnidirectional cameras, multi-projector immersive displays, multi-touch interfaces, and audio processing.

**Seven Evaluated Requirements** From practical experience with our prototypes, we learn that other factors besides eye contact contribute to attaining a true sense of immersivity and copresence in video conferencing. Seven constantly recurring requirements have been identified: eye contact (and the related gaze awareness), spatial context, freedom of movement, visual quality, algorithmic performance, physical complexity, and communication modes (one-to-one, many-to-many, multi-party). We discover that they are subject to many trade-offs and interdependencies as we use them to (informally) evaluate and compare all our prototypes.

A concise sociability study not only points toward the importance of the seven requirements, but also validates our initial preference for image-based methods. However, to arrive at the ideal video conferencing solution, more insight should be gained into the concept of presence, what it means to experience a virtual telepresence and exactly what factors enable this experience. Nevertheless, we believe that the seven requirements provide a reference framework around the experience gained in this dissertation on which to design, develop and evaluate any future solution to eye gaze corrected video conferencing.

# Acknowledgments

This thing has been quite a journey. And a journey like this is never made alone. My gratitude goes out to many people who were there with me along the way.

First and foremost, to my promoter Prof. Dr Philippe Bekaert. Your enthusiasm and love for research is what initially set me on my path. Your vision and guidance prevented me from getting lost, even if you occasionally liked to throw in unexpected turns. Your time and effort, dedicated both to me personally and to the research group as a whole, is greatly appreciated.

Equally invaluable, to my co-promoter Prof. Dr Ir Gauthier Lafruit. In the final stages, your feedback and encouragement kept me on track and helped me reach my destination. I thoroughly enjoyed our discussions on the structure and content of this text, even when I had to tell you (often with a smile on my face) that it couldn't be done in a week. Thank you for not telling me to get it done in a day.

To the members of my jury, for your effort in providing feedback and evaluating my work: Prof. Dr Ir Martin Eisemann, Prof. Dr Ingo Feldmann, Prof. Dr Ir Jacques Verly, Dr Ir Sammy Rogmans, Prof. Dr Wim Lamotte, Prof. Dr Ir Gauthier Lafruit, Prof. Dr Philippe Bekaert, Prof. Dr Frank Van Reeth and Prof. Dr Marc Gyssens. Every good race to the finish needs a jury. Thank you for not disqualifying me.

To all the colleagues I have had the pleasure of working with throughout the years. For the many lively discussions and brainstorming sessions, for co-writing, proofreading and extreme programming, for your mental support while hunting for elusive bugs and solving seemingly unsolvable problems, for your help in constructing numerous demonstrators and arduously setting them up at equally numerous meetings and events, for turning conferences into parties and for joining in the joy of big eureka moments. Or really just for saying hi in the hallways. Research might be about standing on the shoulders of giants, but it has been even more so about leaning on the shoulders of Patrik Goorts, Steven Maesen, Sammy Rogmans, Chris Hermans, Bert De Decker, Cedric Vanaken, Yannick Francken, Tom Haber and many more.

To all the professors and the people behind the scenes who make sure that the EDM remains a well-oiled machine. To Ingrid Konings, Roger Claes, Edith Cloes, Luc Adriaens, Tom De Weyer, Peter Vandoren, Prof. Dr Peter Quax, Prof. Dr Kris Luyten, Prof. Dr Wim Lamotte, Prof. Dr Philippe Bekaert, Prof. Dr Karin Coninx, dean of science, Prof. Dr Frank

Van Reeth, deputy managing director, and Prof. Dr Eddy Flerackers, managing director, for your continuing administrative, technical and managerial support. Your commitment, day in, day out, assures us all of smooth sailing on a toll-free and well-maintained highway.

To friends of old and friends of new, of near and far, inside and outside the EDM. For the support (group), the conversations and the distractions, for the lunch breaks, dinners and barbecues, for the game, quiz and movie nights, for the parties and bar hangouts, for the sportive breaks, for the excursions and field trips, road trips and adventures, for the laughs, pranks (`www.didmaartenfinishhisphdyet.be`, donations are still welcome) and countless more memorable moments and spontaneous shenanigans. Thank you, Ariadna, Ash & Savita, Bart, Chris, David, Devy, Donald, Frederic, Gustavo, Inge, Iva, Jelle, Jeroen, Jordan, Jun, Lode J., Lode & Sofie, Marijke A., Michaël, Nick & Sanne, Patrik, Pavel, Peter B., Sabine, Sammy, Steve, Steven, Stijn, Tim, Thomas, Tom H., and everyone who now feels left out but who has been there in a small or big way. There are simply too many of you, but go say hi to some of you in Figure 2.4, p. 14.

A special thanks is in order to Patrik "pago" Goorts for designing many of the figures in this book, in particular the big black and white overviews. Thanks to you the reader can at least enjoy some worthwhile sightseeing, should he get bored with the text. Another special thanks to Tim "den tikke" De Coster for making the Dutch summary sound less like English and more like actual Dutch. Oh, and for that one note in French…

To Marijke, for the time we have spent together, on paths both through life and through the world.

To Rafaela, for the time we are spending together. For cheering me on and believing in me, for your optimism, your patience and your love. For reminding me what time it is and for calming me down when stress is no longer a choice. I'm looking forward to the time we will be spending together.

And finally, to my family and above all to my parents, Erik en Maria, for always supporting me, always believing in me and always encouraging me. For always being there for me and for giving me all the opportunities in life.

Time to seek out new horizons. See you all down the road.

Thank you,

Maarten Dumont
September 30, 2015

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## Contents

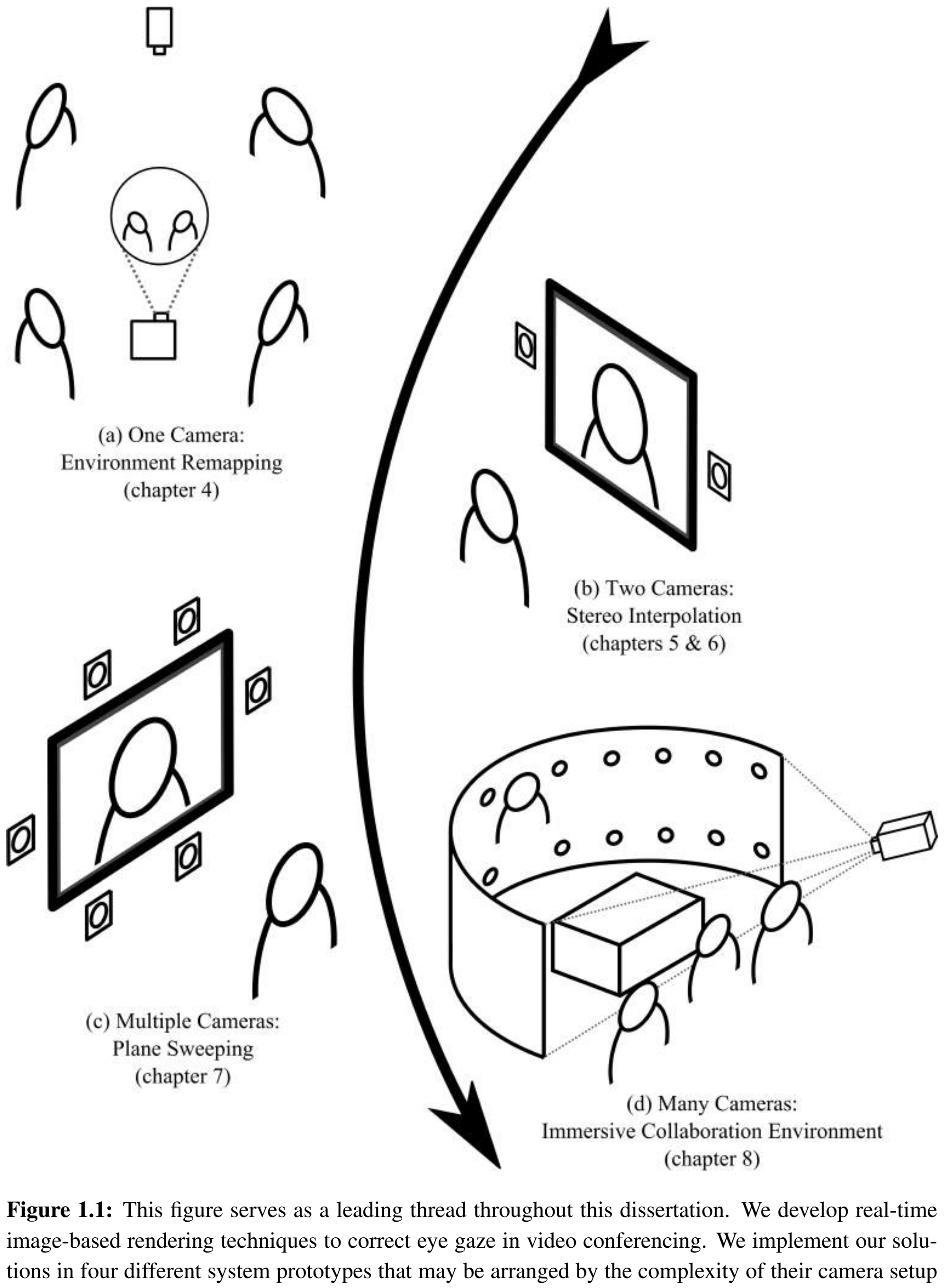


**Figure 1.1:** This figure serves as a leading thread throughout this dissertation. We develop real-time image-based rendering techniques to correct eye gaze in video conferencing. We implement our solutions in four different system prototypes that may be arranged by the complexity of their camera setup at each peer side. We also deal with maintaining the calibration of all the cameras and perform a concise sociability study. Please refer to section 1.2 for a detailed overview.

In today's increasingly globalizing world, economical and ecological factors are making it less and less viable to travel large distances to meet in person. As such, a growing demand for teleconferencing solutions arises to allow people to efficiently and naturally communicate over large distances while simultaneously enjoying a sense of copresence. Audiovisual technology to support this is evolving rapidly, yet many limiting factors prevent a major breakthrough.

## 1.1 Problem Statement

Current video conferencing solutions suffer from some fundamental flaws with regard to attaining a true sense of immersivity and copresence. Consider for a moment the conventional arrangement of a computer screen with a single webcam, shown in Figure 1.2. Eye contact is lost because the user is unable to simultaneously look at the screen and into the camera that is commonly located in the vicinity of the screen. With eye contact missing, the user has to alternate his gaze between the screen to observe his (remote) conferencing partner and the camera to signal his own involvement.

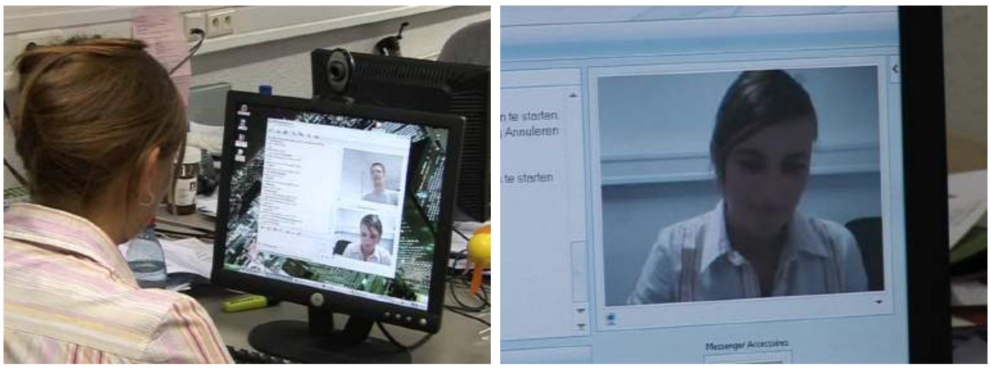


**Figure 1.2:** Standard video conferencing fundamentally suffers from incorrect eye gaze. (left) Direct eye contact is impossible because the user has to choose between looking at his screen or looking into the camera filming him. This causes the disturbing sense that the remote conferencing partner is constantly looking away. (right) Furthermore, only very limited context information on the user's environment is available.

Eye contact (and generally eye expression) has proven to be one of the most crucial elements in human communication and social interaction [Argyle and Cook, 1976; Argyle, 1988; Novick et al., 1996; Cohen et al., 2000; Gemmell et al., 2000]. Serving many purposes, making eye contact influences how people perceive you (e.g. trustworthy or shady, shy or self-confident), asserts dominance or submission, elicits or suppresses response and signals agreement or disagreement (e.g. rolling of the eyes). Broader than strict eye contact, gaze awareness signifies the ability to estimate at who or what (or at least in what direction) a participant in the conversation is gazing. It facilitates the flow control of conversations, specifically with regard to turn taking. Mutual gaze awareness is eye contact as a special case of gaze awareness, where two people are simultaneously aware that they are looking at each other's eyes. Gaze awareness and eye contact may indicate that a participant is ready to speak, ready to listen, wishes to redirect focus or is not attending the conversation anymore.

More limitations apply. A single camera offers only a narrow field of view and therefore provides very little spatial context on the remote user's environment. The lack of context information and the related inability to observe the remote user's actions severely impedes communication and collaboration. A static image also lacks parallax effects that would otherwise create the immersive effect of a virtual window into the world of the remote user.

This dissertation aims to offer solutions to the aforementioned problems and explore their limitations. To this end, we develop and evaluate four different system prototypes. Each prototype takes its own image-based rendering approach, where the correction of eye gaze boils down to the synthesis of a novel view from (live) captured images. Consequently, efficient and accurate view synthesis is where the focus of this dissertation will lie.

Furthermore, as each view synthesis technique is enabled by a specific configuration of the capturing cameras, the prototypes can be arranged according to increasing physical complexity of their camera setup. A more detailed exposition of our prototypes and their view synthesis techniques is coming up in an overview of this dissertation in section 1.2.

From experience with our prototypes, we will discover that seven requirements that any ideal solution should meet will emerge naturally:

- **Eye Contact**: Eye gaze should be corrected, so that the users are able to look each other in the eye. Ideally, this includes the broader concept of gaze awareness, i.e. the more general ability to gauge in what direction, at what or at who the remote user is looking.
- **Spatial Context**: Sufficient spatial context on the environment of the remote user should be made available.
- **Freedom of Movement**: Users should be able to move as freely as possible in their own environment.
- **Visual Quality**: Synthesized eye gaze corrected images should be of high perceptual quality.
- **Algorithmic Performance**: Algorithms should run in real-time with high end-to-end system performance and low latency.
- **Physical Complexity**: The physical construction should be of low complexity, i.e. easy to set up, calibrate and maintain.
- **Communication Modes**: Could the prototype support one-to-one (two-way), many-to-many (two-way) and even multi-party (multi-way) communication?

The last (communication modes) is not a requirement in the strict sense, since it largely depends on what form of communication the prototype is developed to facilitate. We nevertheless include it for ease of comparison, as the form of communication that we choose to support may influence the other requirements. Furthermore, the requirements are subject to many trade-offs and interdependencies that impact the user experience. For example, offering a higher visual quality will demand more computationally intensive algorithms, which in turn may decrease the end-to-end performance of the system. It is challenging to optimize for all requirements simultaneously and we will observe these issues in practice.

For each prototype that we present, the requirements will be evaluated on a 7-point scale, informally defined as:

| | |
|---|---|
| 1 | Terrible |
| 2 | Bad |
| 3 | Reasonable |
| 4 | Average |
| 5 | Good |
| 6 | Very Good |
| 7 | Excellent |

This will allow us to compare the sometimes very diverse prototypes in a consistent manner. The evaluations can be found at the end of each prototype's dedicated chapter, in a section specifically titled *Requirements Evaluation*, i.e. in section 4.7, p. 63, in section 6.5, p. 124, in section 7.5, p. 154, and in section 8.5, p. 180. They give a clear and concise overview of each prototype's strengths and weaknesses.

## 1.2 Dissertation Overview

We classify our prototypes for eye gaze corrected video conferencing by the complexity of the camera setup at each peer side, starting at a very minimum and working our way up to a multitude of cameras and associated increased processing. In this way, as the complexity of its camera setup increases, the requirements summarized in section 1.1 that the prototype is able to fulfill will also vary. It is this classification that dominates the structure of this dissertation. It has been conceptualized in Figure 1.1, which will serve as a leading thread.

In chapter 2, we offer background on the main topics covered in this dissertation. Previously proposed solutions and related techniques will be explored and placed into context.

Before we can get the ball rolling, we must first learn how to keep cameras calibrated in an environment that can be subject to a lot of dynamic user activity. In chapter 3, we introduce our image-based approach to maintaining a functional camera calibration in an a priori calibrated network of cameras [Hermans et al., 2007b].

In chapter 4, we present our first prototype. It requires only the very minimum amount of cameras, that is to say a single one. Based on the concept of environment mapping, it combines the capture of omnidirectional video by filming a hemispherical mirror with corresponding projection on a hemispherical screen [Hermans et al., 2006, 2007a]. Both capture and display are performed by a single camera (resp. projector) on a single sphere, forming the single communication device depicted in Figure 1.1(a). The users are located in the 360-degree space around the sphere, thereby receiving nearly unlimited freedom of movement and spatial context. However, the visual quality suffers and the physical setup is unconventional at best.

In chapter 5, we develop a fast and accurate novel stereo matching algorithm [Dumont et al., 2014b,c, 2015]. This algorithm is then applied to the problem of eye gaze correction in chapter 6, where we delve into rectified stereo interpolation to synthesize an eye gaze corrected image from two cameras mounted to the left and to the right of the screen. This is depicted in Figure 1.1(b). Although this is imaginably the least complex physical setup and has superior visual quality, we expect it to offer fairly limited freedom of movement and arguably little spatial context.

In chapter 7, we attempt to offer a higher degree of freedom of movement and smoother overall performance by mounting more than two cameras around the screen. Six cameras are used in practice, as represented in Figure 1.1(c). As such a general camera configuration can no longer be processed by conventional rectified stereo algorithms, we turn to plane sweeping to interpolate the eye gaze corrected image [Dumont et al., 2008, 2009b; Rogmans, 2013; Dumont et al., 2014a]. We will see that plane sweeping inherently allows for much more flexibility in choosing a viewpoint to maintain eye contact.

We present the final logical evolution in chapter 8, where we aim to achieve the would-be holy grail of immersive video conferencing and, consequently, collaboration environments. Specifically, we drastically improve the user experience by projecting the environment on a spatially immersive display, together with enabling collaboration by providing both sides with networked surface computing, as illustrated in Figure 1.1(d) [Dumont et al., 2010, 2011]. Not only does this prototype at its core lean on the techniques and algorithms developed in the previous chapters, but the observant reader will also recognize that many key research areas that have been the focus of our research group as a whole over the past years are truly brought together. The result is a very high level of immersivity, coupled with a broad freedom of movement and support for both one-to-one and, ultimately, many-to-many use cases.

In chapter 9, we pose the important question of whether users really require eye contact when communicating in the first place. We look for the answer to this and other questions in a concise sociability study.

Finally, we reflect and conclude in chapter 10.

## 1.3 Scientific Contributions and Acknowledgments

We summarize our main research contributions. We also give scientific acknowledgment by citing all original publications, because progress is rarely made alone.

- In chapter 3, we develop a novel algorithm to keep cameras calibrated in a dynamic environment. A displaced camera is reintegrated into the initially calibrated camera network by tracking salient features in the images and recomputing the extrinsic calibration based solely on image-space correspondences. [Hermans et al., 2007b]
- We develop and compare three different view synthesis methods to compute an eye gaze corrected image: environment remapping in chapter 4, rectified stereo matching and interpolation in chapters 5 & 6, and plane sweeping in chapters 7 & 8.
- We develop four different prototype systems that rely on our three different view synthesis methods to reconstruct the eye gaze corrected image:
  - For our first prototype in chapter 4, we capture incoming rays of light on one specular side of a sphere and develop the equations that map them to outgoing rays of light on the other diffuse side of the sphere. The resulting image transformation is completely independent of the scene depth and is placed in the context of environment mapping. [Hermans et al., 2006, 2007a]
  - For our second prototype in chapter 6, we investigate the limits of rectified stereo interpolation in the context of close-up one-to-one eye gaze correction. Also, the conventional depth-image-based rendering pipeline is refined and extended. Rectified stereo interpolation requires accurate disparity estimation:
    - In chapter 5, we develop a novel disparity estimation algorithm. First, we aggregate matching costs using edge-sensitive windows that adapt their shape in both a horizontal and vertical orientation. Second, we formalize an iterative disparity refinement process that can be applied to any local disparity estimation algorithm and that has a large effect on the final quality of the disparity map. [Dumont et al., 2014b,c, 2015]
  - In chapter 7, we rely on plane sweeping to develop our third prototype: an end-to-end system prototype for close-up one-to-one eye gaze corrected video conferencing. Plane sweeping allows us to efficiently interpolate any viewpoint between multiple cameras closely surrounding the screen. We combine this with an eye tracker to continuously determine the optimal position of the virtual camera and thereby avoid the horizontal baseline restriction of rectified stereo interpolation. Each module is carefully defined for optimal system performance, even on low-end hardware. We investigate the end-to-end system performance. [Dumont et al., 2008, 2009b,a; Rogmans, 2013; Dumont et al., 2014a]
  - In chapter 8, we bring all of this together in our fourth and final prototype, an immersive collaboration environment, by implementing and extending the fundamental requisites of *The Office Of The Future* defined by Raskar et al. [1998]. [Dumont et al., 2010, 2011]

- We develop several dynamic control mechanisms for stereo matching and plane sweeping that decrease their algorithmic complexity while increasing their view synthesis quality. The control mechanisms are all based on a histogram analysis of the concerning disparity or depth map. They are developed for stereo matching by hierarchically restricting the disparity range (in section 6.2, p. 105) [Rogmans et al., 2009a; Rogmans, 2013], for single object-of-interest plane sweeping by redistributing the planes uniformly in a dynamic depth range (in section 7.2.5, p. 142) [Dumont et al., 2008, 2009b; Rogmans, 2013; Dumont et al., 2014a] and for full-scene plane sweeping by redistributing the planes non-uniformly in a fixed depth range (in section 8.3, p. 170) [Goorts et al., 2013b; Goorts, 2014; Goorts et al., 2014b; Dumont et al., 2014a].

- We design all our view synthesis methods for – and implement them on – the GPU to take advantage of its massive parallel processing infrastructure. In fact, this whole dissertation from beginning to end runs on the GPU, an approach known as general-purpose GPU (GPGPU) computing.

- In section 1.1, we define seven requirements that any solution for immersive video conferencing should meet: eye contact (and the related gaze awareness), spatial context, freedom of movement, visual quality, algorithmic performance, physical complexity, and communication modes. We observe these requirements to naturally emerge from experience with our prototypes.

- For each of these requirements, we (informally) evaluate our four prototypes on a 7-point scale at the end of their dedicated chapters. These evaluations give a clear and concise overview of each prototype's strengths and weaknesses.

- In chapter 9, we perform a sociability study on eye gaze correction. Although concise, the study does provide pointers in the right direction. [Mechant et al., 2008; van Nimwegen, 2008]

## 1.4 Demonstration Videos

Videos that demonstrate our various solutions to eye gaze correction in video conferencing are available online. We invite the reader to have a look in order to gain a better understanding of their possibilities and complexities. Please visit any of:

- `http://research.edm.uhasselt.be/~mdumont/phd/`

- `http://phd.maartendumont.net`

- `http://www.youtube.com/user/dwerftje/`

# Chapter 2

# Background and Related Work

## Contents

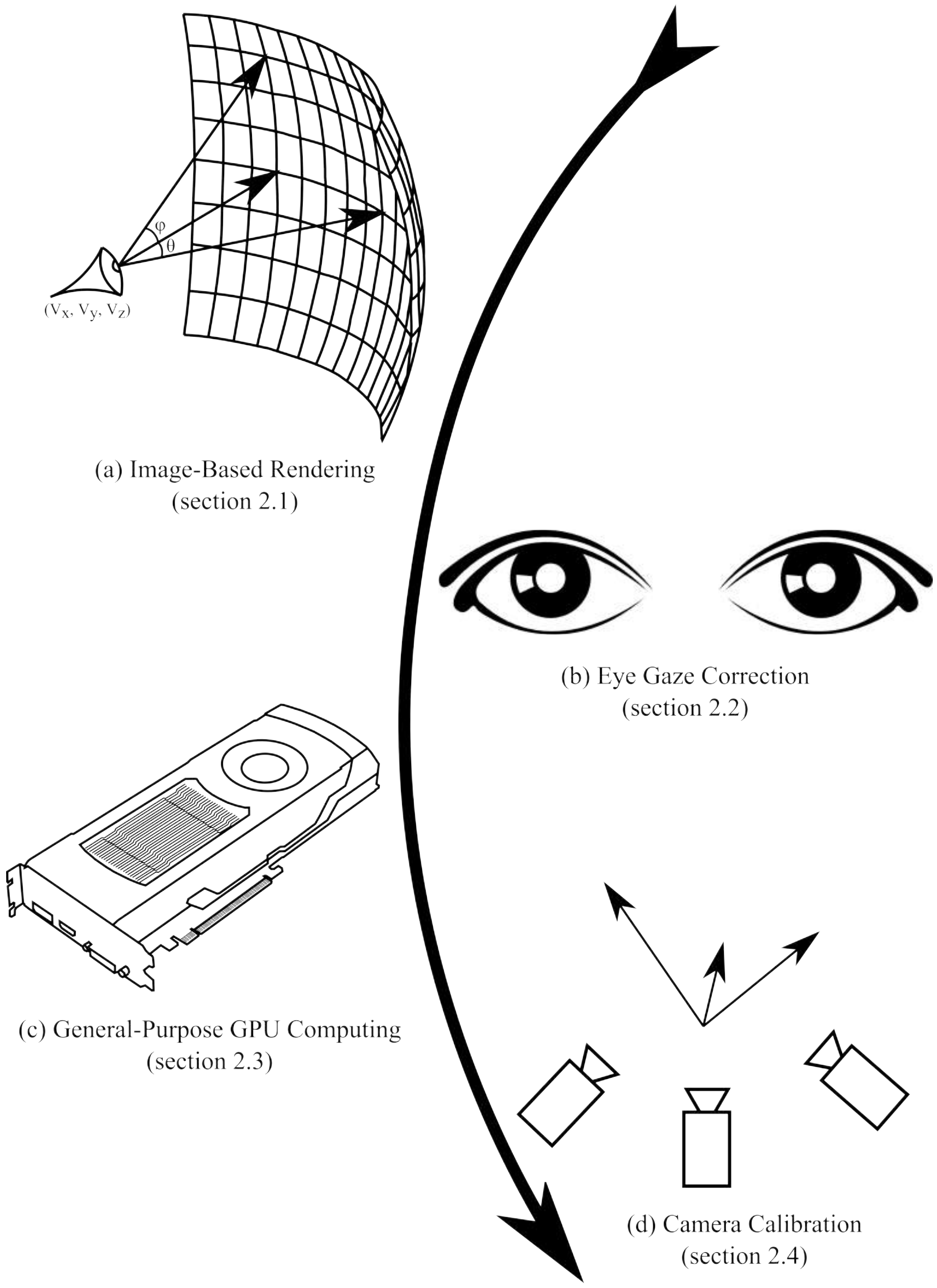

φ
θ
(Vx, Vy, Vz)
(a) Image-Based Rendering (section 2.1)
(b) Eye Gaze Correction (section 2.2)
(c) General-Purpose GPU Computing (section 2.3)
(d) Camera Calibration (section 2.4)

**Figure 2.1:** This chapter offers background on the main topics covered in this dissertation.

This chapter offers background on the main topics covered in this dissertation, summarized in Figure 2.1. The reader who is familiar with these topics may skip this chapter. We will refer when relevant in the following chapters.

*View interpolation* is the first part of this dissertation's title. Most approaches to eye gaze correction boil down to the interpolation of a novel view from images acquired by any number of cameras in a configuration under certain constraints. We will investigate two of these view interpolation methods, more specifically rectified stereo interpolation in chapters 5 & 6 and plane sweeping in chapter 7. In chapter 4, however, we will first demonstrate that the same eye gaze correcting effect can be achieved without conventional interpolation by capturing the environment from, and subsequently reprojecting it on, appropriately curved rather than planar surfaces. All of these view synthesis methods can be placed in the broader context of image-based rendering in section 2.1.

We use view interpolation to solve the problem stated in the second part of this dissertation's title: *correcting eye gaze in video conferencing*, discussed in section 2.2.

All view synthesis methods are designed for and implemented on the GPU to take advantage of its massive parallel processing infrastructure. In section 2.3 we briefly explain how to exploit the GPU for general-purpose, i.e. non-graphics, computations and thereby achieve the *real-time* keyword of this dissertation's title.

But before getting there, all of our solutions ask for accurately calibrated cameras. Maintaining the calibration of an a priori calibrated camera in a dynamic environment will be the topic of chapter 3. Some background on initial camera calibration is given in section 2.4.

## 2.1 Image-Based Rendering

Conventional computer graphics requires an a priori specified 3D geometric model of the scene (by means of e.g. vertex meshes, polygon triangulations, sub-division surfaces, etc.), together with color, texture, lighting, etc. This data is then processed by the rendering pipeline which ultimately performs a perspective projection of the 3D model to produce a 2D rendered image. Image-based rendering (IBR), on the other hand, does not start from a common geometric model. Instead, it synthesizes a novel view directly from real-world captured images. The advantage is a built-in degree of true-to-life visual realism that is difficult to achieve with traditional geometric modeling and texturing.

Although many IBR methods often defy rigid classification, they can still be broadly categorized on a spectrum that requires little to no geometry on one end (section 2.1.1) and explicitly provided geometry on the other (section 2.1.3), with varying degrees of implicitly determined geometry in between (section 2.1.2). Usually a trade-off is observed between (the accuracy of) the geometry and the number of images: the less geometry, the more input images (from differing viewpoints) are required to reliably reconstruct a scene, and vice versa. The spectrum is visualized in Figure 2.2. For an extensive overview, please refer to Shum and Kang [2000], Magnor [2005], Shum et al. [2007] and Szeliski [2010].

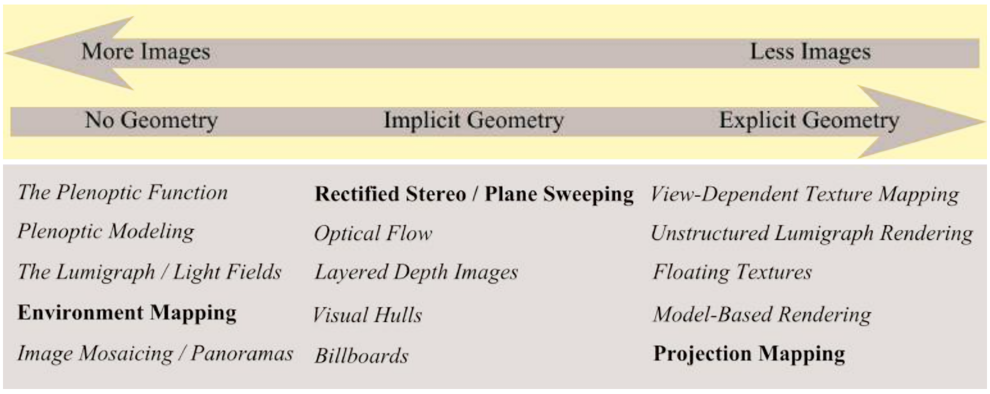


**Figure 2.2:** Image-based rendering (IBR) algorithms can be broadly categorized on a spectrum that requires little to no geometry on one end and explicitly provided geometry on the other, with varying degrees of implicitly determined geometry in between. A geometry versus number of images trade-off is observed. Environment mapping and projection mapping are used in chapter 4, rectified stereo in chapters 5 & 6 and plane sweeping in chapter 7. Spectrum adapted from Szeliski [2010].

### 2.1.1 IBR Without Geometry

Image-based rendering techniques that do not require any geometric information about the scene all rely on some (more or less restricted) form of the plenoptic function.

**The Plenoptic Function** The plenoptic function in its most general form was originally conceived by Adelson and Bergen [1991] as the 7D function of the intensity of light rays arriving at every viewing location $(V_x, V_y, V_z)$, at every possible angle represented by the spherical coordinates $(\theta, \phi)$ and for every wavelength of light $\lambda$ at every time $t$:

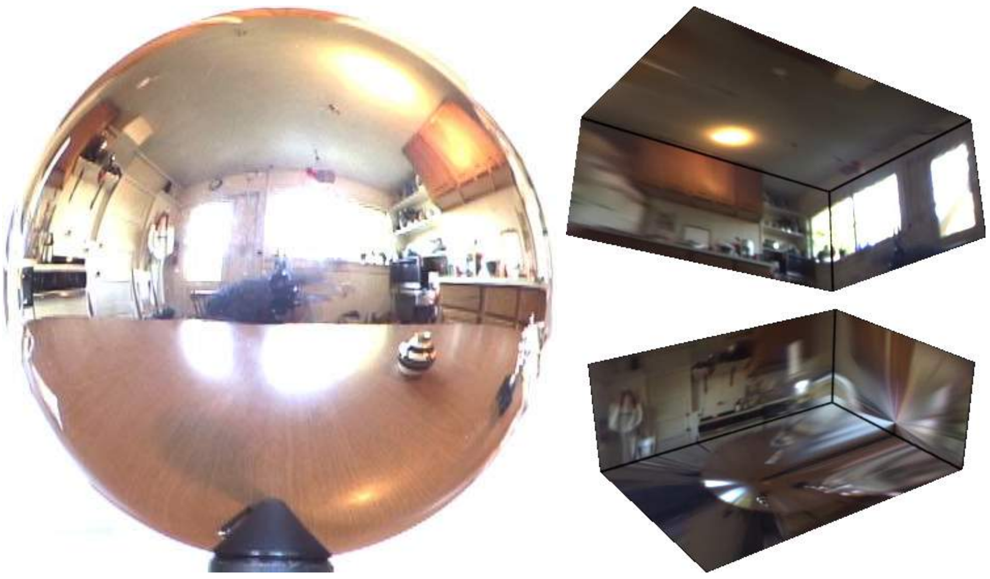
$$P_7 = (V_x, V_y, V_z, \theta, \phi, \lambda, t)$$

This all-encompassing function in essence answers the question: what information about the world is contained in the light filling a region of space at any time?

**Plenoptic Modeling** McMillan and Bishop [1995] define a 5D sample $P_5$ of the general plenoptic function $P_7$ to be the full spherical map of a scene at a fixed moment in time:

$$P_5 = (V_x, V_y, V_z, \theta, \phi)$$

where the wavelengths $\lambda$ from $P_7$ collapse into the RGB channels of a recorded color image. An incomplete sample is some solid angle subset $(\theta, \phi)$ of this spherical map, shown in Figure 2.1(a). They define plenoptic modeling to be the goal of reconstructing a novel viewpoint of a scene from reference images (i.e. incomplete 5D samples) of that scene.



(a) Environmental reflection of a residential kitchen on a specular sphere.

(b) Unfolded on the sides of a box.

**Figure 2.3:** Concept of environment mapping: unfolding (a) the environmental reflection captured on a specular sphere yields (b) omnidirectional imagery with the projection center located at the center of the sphere Debevec [1998].

**The Lumigraph and Light Fields** If we stay outside an object's bounding box (or more precisely, its convex hull), the 5D plenoptic function simplifies to a 4D function $P_4$:

$$P_4 = (u, v, s, t)$$

where $(u, v)$ and $(s, t)$ parametrize two planes parallel to the bounding box. This 4D function was concurrently formalized as the lumigraph by Gortler et al. [1996] and as light field rendering by Levoy and Hanrahan [1996]. To capture a light field, the position of the camera is restricted to the $(u, v)$ plane, with its focus on the $(s, t)$ plane. To reconstruct a novel viewpoint, the radiance information from both planes is resampled. Jorissen et al. [2014] qualitatively compare light field rendering with the MPEG stereo interpolation algorithm.

**Environment Mapping** The more constraints we put on the camera location, the simpler the plenoptic function becomes. If we do not move at all, we end up with a 2D spherical environment map:

$$P_2 = (\theta, \phi)$$

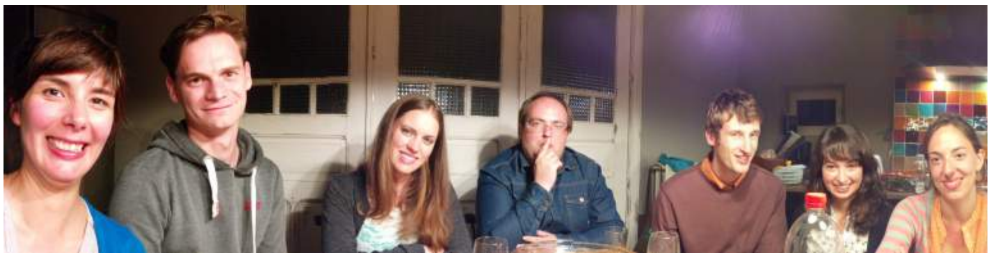


**Figure 2.4 (continued on facing page):** A panorama is a solid angle subset of an environment map. An international group of friends are enjoying each other's company around a dinner table. From left to right: Rafaela (PT), Donald (BE), Jordan (US), Steven (BE), Lucas (FR), Blanca (ES), Sofie (BE), Inge (BE), Iva (BG), Gustavo (MX), Sabine (BE), Patrik (BE) and Maarten (BE, author). July 26, 2015.

which captures all light that arrives from all directions at a fixed point in the scene.

Debevec [1998] captures environment maps by filming a (relatively small) specular reflective sphere in the center of a (relatively distant) scene. As he demonstrates in Figure 2.3, unfolding the captured environmental reflection yields omnidirectional imagery with the projection center located at the center of the sphere. This observation is key to our solution for eye gaze correction in chapter 4, as we will capture environment maps over time (in other words, video) by filming a spherical mirror, thus adding back the time parameter: $P_3 = (\theta, \phi, t)$.

**Image Mosaicing and Panoramas** For a solid angle subset, $P_2$ reduces to image mosaicing, i.e. capturing a sequence of images along a predefined viewpoint, as we did in Figure 2.4. Many approaches have been proposed to construct cylindrical and spherical panoramas by stitching multiple images together, among them by Chen [1995], Szeliski [1996] Hermans et al. [2008] and Vanaken [2011].

Capturing panoramas and environment maps is even easier if fisheye lenses [Xiong and Turkowski, 1997] or omnidirectional cameras [Nalwa, 1996; Nayar, 1997; Peleg et al., 2001; Weissig et al., 2012] are used. We will use a Point Grey Ladybug3 omnidirectional camera to capture the user's panoramic background in our immersive collaboration environment in chapter 8.

## 2.1.2 IBR With Implicit Geometry

Image-based rendering with implicit geometry first estimates the depth of the scene from any information contained in the input images. Once depth is available for every pixel of an image, the image can be rendered from any nearby viewpoint by warping its pixels to the image plane of the desired viewpoint, based on the recovered depth information [McMillan, 1997]. This class of image-based rendering algorithms is therefore also known as depth-image-based rendering (DIBR).

**Stereo Matching and Interpolation** In Figure 2.5(a), two objects are placed at different depths in front of a pair of cameras. When switching from the left to the right camera position, the objects' position relative to each other (and to the background) will appear to change. Objects in the background (the palm tree) will appear to move less in comparison to objects in the foreground (the blue buddy). This effect is known as the parallax effect.

If both cameras are in a perfectly rectified stereo configuration (i.e. if their epipolar lines run parallel with the X-axis), the parallax effect causes a horizontal displacement of the pixels in both captured images. This displacement of a pixel is called the pixel's disparity (see Figure 2.5(b)) and is inversely proportional to the object's depth in the scene:

$$d = f \times \frac{b}{z} \tag{2.1}$$

with $d$ the disparity, $z$ the depth, $f$ the camera's focal length measured in pixels and $b$ the baseline distance [Chai et al., 2000].

The goal of stereo matching is to compute a dense disparity map by estimating each pixel's displacement [Scharstein and Szeliski, 2002], which will be the topic of chapter 5. The disparity map can then be used to synthesize intermediate viewpoints by shifting each pixel on its scanline proportionally to its disparity [Scharstein, 1996; Rogmans et al., 2009c]. For example, in Figure 2.5(c) each pixel of the left color image was shifted with half its disparity to synthesize a novel image positioned exactly halfway between the left and the right camera viewpoints. Though heavily simplified here, in essence this is how we will correct eye gaze in chapter 6.

**Plane Sweeping** Take a look at Figure 2.6 to conceptually grasp plane sweeping [Collins, 1996; Yang et al., 2002]. Cameras $C_1$ to $C_3$ are real cameras in a general configuration, whereas camera $C_v$ is the virtual camera of which we wish to reconstruct the color image and the scene depth.

Any voxel $f$ in 3D space at e.g. plane depth $z_1$ is projected onto the pixels $p_i$ on the 2D image plane of the respective cameras $C_i$. Conversely, inversely projecting (i.e. deprojecting) the pixels $p_i$ into 3D space will have them match at one single voxel $f$ at the corresponding plane depth $z_1$. On all other plane depths $z_j$ ($j \neq 1$) the pixels $p_i$ deproject on points $f_i$ that do not coincide, as illustrated on plane depth $z_2$. Visually, reprojecting these points $f_i$ back

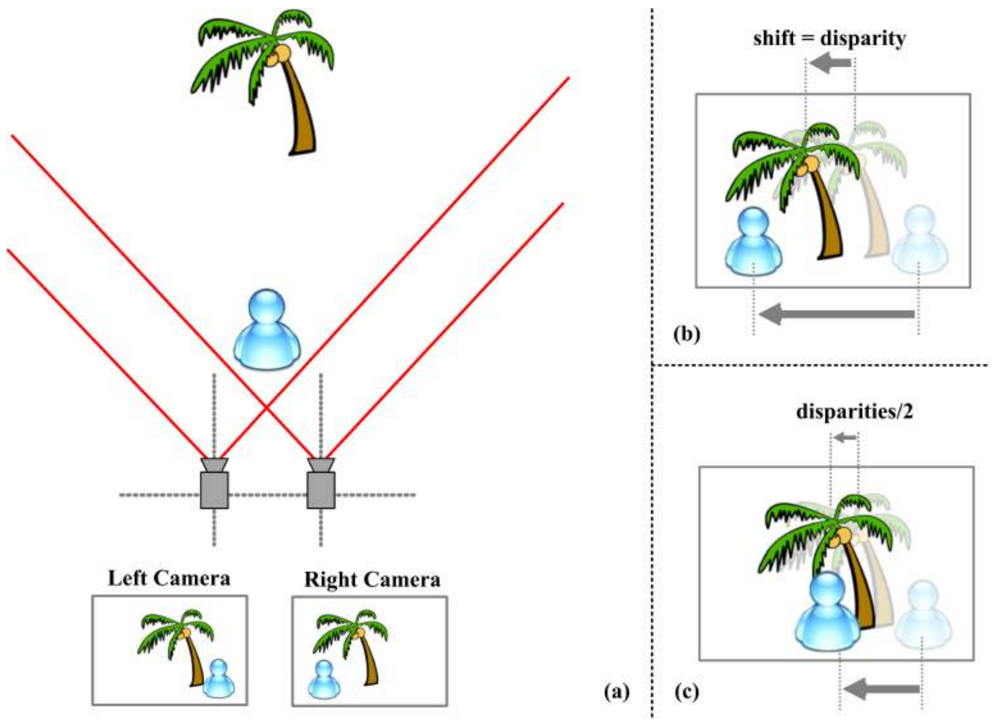


**Figure 2.5:** Concept of stereo matching and interpolation. (a) A scene is captured using two rectified cameras. Stereo matching attempts to estimate the apparent movement (i.e. disparity) of the objects across the images. (b) A large disparity corresponds to close objects (a low depth value) and vice versa. (c) If the disparity of each pixel is known, intermediate viewpoints can be interpolated.

onto the image plane of the virtual camera $C_v$ causes a non-focused ghosting artifact on the virtual image plane.

For instance, for the two person scene of Figure 2.6, the person answering the phone at the desk in the foreground will be in focus at depth $z_1$ (see projected images $Iz_1$), whereas the person walking by in the background is out of focus at the same hypothesized depth $z_1$. The background person in turn will be in focus on plane depth $z_2$ (see projected images $Iz_2$), hence suggesting that his corresponding voxels are indeed at depth $z_2$. Looking even deeper into the scene, the whiteboard in the far background is de- and reprojected in focus at its corresponding plane depth, say $z_4$, and is now in fact readable (see projected images $Iz_4$).

Plane sweeping, used to correct eye gaze in chapter 7, can be regarded as a more generalized form of stereo matching. They are closely related techniques that estimate the depth-of-scene from image correspondences, determined using some measure of (color-based) consensus between the pixels. By design, plane sweeping processes multiple input images simultaneously and is thereby more robust to illumination mismatches and camera misalignment.

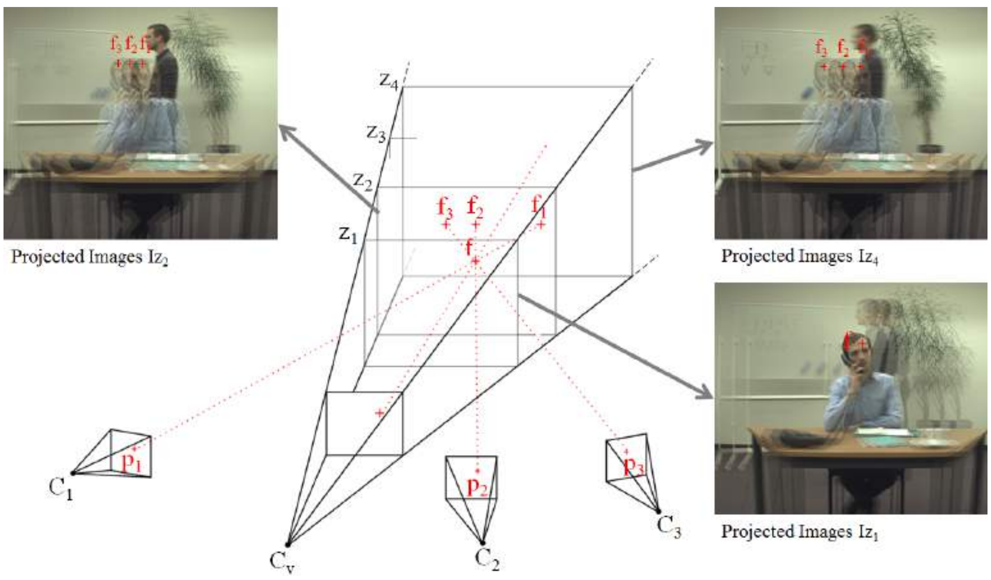


**Figure 2.6:** Concept of plane sweeping. Deprojecting real cameras $C_i$ ($1 \leq i \leq 3$) on different plane depths $z_j$ for virtual camera $C_v$ may cause ghosting, depending on whether or not the scene object in question is present at depth $z_j$.

Also unlike stereo matching, which requires its input image pair to be stereo rectified, plane sweeping inherently rectifies its input images by projecting them onto the planes. Furthermore, the relation between plane depths $z$ and stereo disparities $d$ is again expressed by Equation 2.1. Finally, the recovery of the scene depth is not essential in plane sweeping, but rather a by-product of a novel view rendering process. Even so, if nothing else, the reconstructed depth can be helpful during image post-processing.

**Optical Flow** Stereo matching assumes that the images are stereo rectified so that the search for pixel correspondences is restricted to corresponding scanlines. Optical flow, also known as motion estimation, relaxes this condition by widening the search for pixel correspondences to any 2D direction on the image planes [Lucas and Kanade, 1981; Shi and Tomasi, 1994]. If the epipolar geometry is known, the epipolar line corresponding to a pixel in one image can constrain the search for corresponding pixels in the other image. From two input images, given dense optical flow between them, the view interpolation method of Chen and Williams [1993] can reconstruct arbitrary viewpoints. It can be seen as a relaxation of rectified stereo to tensor space, which expresses the relation between three images in a generalized configuration [Avidan and Shashua, 1997].

**Layered Depth Images** Information in the synthesized image may be missing because it was hidden behind (occluded by) foreground objects in the reference image. Layered depth images alleviate this problem by storing several depth and color values for every pixel in a reference image, especially for pixels near transitions between foreground and background [Shade et al., 1998; Chang et al., 1999; Zitnick et al., 2004]. Layers can be hierarchically composed from a larger collection of images using optical flow.

**Visual Hulls** As we get closer on the spectrum in Figure 2.2 to methods that rely on explicitly defined geometry, we move away from depth maps defined in 2D image space to – still implicitly determined – geometry defined in 3D world space. We first encounter the well-known concept of visual hulls, introduced by Laurentini [1994]. A foreground object's silhouette (i.e. segmentation) is deprojected to a cone in 3D space. The intersection of multiple of these cones from multiple cameras defines the object's visual hull. The visual hull is an equal or tighter fit than the object's convex hull and is silhouette-consistent across the cameras, meaning it projects within the object's silhouette in each camera. Matusik et al. [2000] describe an efficient image-based approach to computing and texturing visual hulls that are reconstructed from cameras in a wide-baseline configuration. The concept was further refined as photo hulls (taking into account color information) by Slabaugh et al. [2002] and as depth hulls (from active depth cameras) by Bogomjakov et al. [2006]. [Feldmann et al., 2009a] integrate visual hulls and stereo matching to estimate depth for their 3D multi-user video conferencing system.

**Billboards** Lastly, billboards [Hayashi and Saito, 2006; Waschbüsch et al., 2007] are straightforward geometric proxies (usually planes) that are placed in 3D world space at the object of interest's general location, yet are always oriented toward the viewing camera. A novel viewpoint is reconstructed by rendering the proxies, textured with the appropriate color information. While perspective distortions may be perceived by projecting on too simplified geometry proxies, the visual quality remains reasonable when keeping the novel viewpoint close to the reference camera positions.

### 2.1.3 IBR With Explicit Geometry

Traditional computer graphics maps textures on predefined geometry. It requires very accurate geometric models but only a sparse set of color images (textures).

**View-Dependent Texture Mapping** View-dependent texture mapping was introduced by Debevec et al. [1996] as a method of texturing a more basic model of the scene, while still simulating geometric detail. They incorporate a stereo matching algorithm to determine the deviation between the real scene and the basic model. Their stereo matching relies on the basic model to robustly recover accurate depth from more widely-spaced image pairs. They later demonstrated how to efficiently implement view-dependent texture mapping on graphics hardware by exploiting its projective texture mapping capabilities [Debevec et al., 1998].

**Unstructured Lumigraph Rendering** Unstructured lumigraph rendering [Buehler et al., 2001] combines the principles of light field rendering with those of view-dependent texture mapping and is thereby able to relax (approximate) the geometry proxy. Unlike light field rendering, the cameras and image planes are no longer restricted to be coplanar. Multiple cameras are blended together according to the angle of the view rays, field of view and resolution penalties, and visibility (occlusions) in relation to the geometry proxy. Take a look ahead at Figure 7.16, p. 152, showing an unstructured lumigraph rendering of a user in front of our prototype for eye gaze correction that employs plane sweeping to reconstruct the geometry proxy.

**Floating Textures** More recently, Eisemann et al. [2008] apply floating textures to coarse geometry to generate novel viewpoints. To cope with 3D geometry errors and visibility constraints, they allow textures to *float* over the geometry, while ensuring projected textures are color consistent by rearranging parts of the textures using optical flow methods.

**Model-Based Rendering** Full model-based rendering reconstructs a model of the (human) actors in the scene, often with the goal of synthesizing not only new viewpoints [Carranza et al., 2003; Starck and Hilton, 2003], but new poses too [Vanaken et al., 2008; Vanaken, 2011].

**Projection Mapping** Projection mapping leaves the virtual world and enters the physical one by projecting on real-world objects of various shapes and sizes. To ensure correct perspective, it requires an accurate geometric model of the object [Raskar et al., 2001, 2003]. The field is also known as spatial augmented reality [Bimber and Raskar, 2005]. In chapter 4 we will illuminate a spherical display as if it were a reflective mirror. In chapter 8 we will use multiple projectors to cover a 180-degree panoramic screen in our immersive collaboration environment.

## 2.2 Eye Gaze Correction

Early solutions to correcting eye gaze took to model-based approaches. Vetter [1998] maps a single image of a face on a generic 3D model of a human head. Similarly, Jerald and Daily [2002] warp the imagery of the eyes within a single captured video frame so that the eyes appear to be looking at the camera, even though the head is turned away. Gemmell et al. [2000] texture map the face onto a 3D head model and furthermore replace the eyes with newly synthesized ones with corrected gaze. Yang and Zhang [2002] triangulate a 3D model from point correspondences determined by stereo matching and then warp the model to the correct virtual viewpoint orientation. Giger et al. [2014] apply recent shape deformation techniques to generate a 3D face model that matches the user's face. They then render a gaze corrected version of this face model and insert it into the original image. Schreer et al. [2008a] do away with all image-based rendering and instead track the users' facial expressions and gestures and map those to animated avatars. Similarly, Di Fiore et al. [2008] let facially animated avatars convey emotions in networked virtual environments. Generally, these model-based solutions often lack in natural expression, with users talking to humanoid looking avatars. The visual quality is directly dependent on the quality of the model and we will therefore not investigate these methods further.

One of the first purely depth-image-based rendering approaches is presented by Criminisi et al. [2003]. They employ stereo matching to estimate dense disparity maps from rectified images captured by two cameras placed on either side of the screen. They then achieve eye contact by synthesizing a view from a virtual camera that is located roughly where the image of the head should be displayed on the screen. We will take a similar approach to eye gaze correction in chapter 6. However, their stereo matching algorithm is based on dynamic programming, which typically suffers from a *streaking* effect between horizontal scanlines. Furthermore, they implement their framework on commodity CPUs, resulting in a very low frame rate when sufficient visual quality is required. In chapter 5 we develop a novel stereo matching algorithm that is able to generate high quality disparity maps in real-time by massive parallel processing on the GPU.

Recent research has paid attention to image-based rendering techniques that integrate geometry acquired by consumer-level active depth cameras such as the Microsoft Kinect [Zhang, 2012]. Kuster et al. [2012] synthesize only the gaze corrected face and transfer it in a seamless manner into the original image. Maimone et al. [2012] merge data acquired from multiple depth cameras and present techniques for automatic color calibration. We will also use the Kinect to guide the view interpolation when multiple users are in view in our immersive collaboration environment in chapter 8.

Opposite to image-based rendering approaches, hardware-based solutions often involve the use of expensive dedicated hardware or an unpractical system setup. Vertegaal et al. [2003] approximate eye contact by placing three cameras behind a half-silvered mirror and selecting the camera that is closest to where the user is looking. On the other hand, in an attempt to offer a higher degree of freedom of movement, the Coliseum system by Baker et al. [2002, 2005] features five cameras in an unpractical 180-degree configuration surrounding the user at eye level.

Many others optimize parts of the application, such as multi-view video coding [Chien et al., 2003; Guo et al., 2005] for efficient data communication, or the networking component and QoS in multi-party settings [Yang et al., 2005, 2006b,c; Wu et al., 2008; Yang et al., 2010b]. However, neither of them integrate and optimize the entire end-to-end system. In chapter 7 we will develop a fully functional prototype for close-up one-to-one eye gaze corrected video conferencing and analyze its end-to-end performance.

## 2.3 General-Purpose GPU Computing

An everyday part of consumer-grade personal computers, the graphics processing unit (GPU) is responsible for rendering a 3D scene to a 2D display. It can, however, also be put to work to solve problems other than graphics rendering, a concept known as general-purpose GPU (GPGPU) computing [Luebke and Humphreys, 2007; McCool, 2007; Owens et al., 2007, 2008; Asanovic et al., 2009].

Compared with a CPU's SISD (single instruction, single data) design, the GPU's SIMD (single instruction, multiple data) architecture offers much higher throughput and scalability, at the expense of caching and flow control. These properties are especially advantageous to algorithms that are easily parallelizable: in particular all image processing in this dissertation, but also searching, sorting, algebraic problems, signal processing, physics simulations and many more [Fernando, 2004].

The original approach to GPGPU was to trick the GPU into making general-purpose computations by reprogramming its graphics rendering pipeline (section 2.3.1). Being a pragmatic – but nevertheless valid – approach to GPGPU, it is clear, however, that the GPU was originally not designed for this purpose. The graphics API is misused and intricate knowledge of the rendering pipeline and its hardware implementation is required. To overcome these hurdles and to ease the adoption of GPGPU, in recent years the GPU has been transformed from a pure graphics renderer into a general massively parallel computing platform (section 2.3.2). Even so, when developing our various view interpolation algorithms for eye gaze correction, both approaches maintain their specific strengths and weaknesses [Ryoo et al., 2008; Rogmans et al., 2009b; Goorts et al., 2009, 2010].

### 2.3.1 The Programmable Graphics Pipeline (OpenGL/Cg)

To render a geometric model of a scene to the screen, the GPU follows the graphics rendering pipeline outlined in Figure 2.7. This pipeline consists of a number of stages that operate in parallel, while at the same time the output of each stage serves as sequential input for the next stage [Shreiner, 2009].

Input to the pipeline is provided by the host application in the form of ordered vertex data. This data must minimally contain the vertices' positions (homogeneous quadruplets in world space) and primitive assembly information. The assembly information describes how distinct vertices combine to form the boundaries of geometric primitives, i.e. basic drawing shapes such as points, lines and (most commonly) triangles that compose the models in the scene. Optional property information for each vertex may be included, such as texture coordinates, color values, lighting components, normal vectors, etc. Lastly, the graphics pipeline is (to a limited extend) configurable by setting its various state parameters, e.g. the position and orientation of the viewing camera, the view frustrum (the visible volume of the scene in front of the viewing camera), the viewport (the output area in the framebuffer), etc.

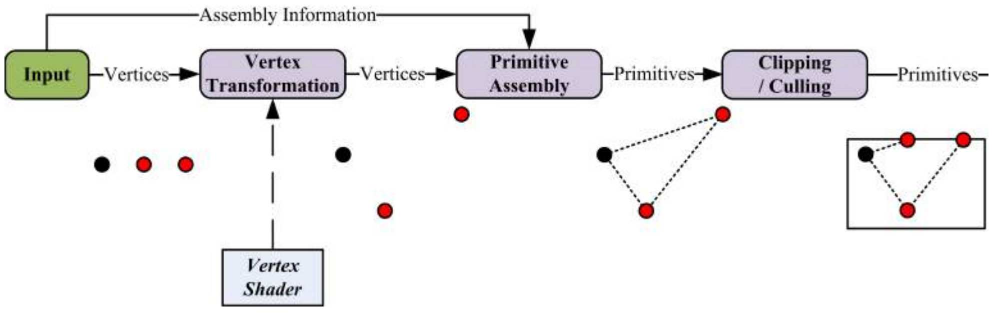


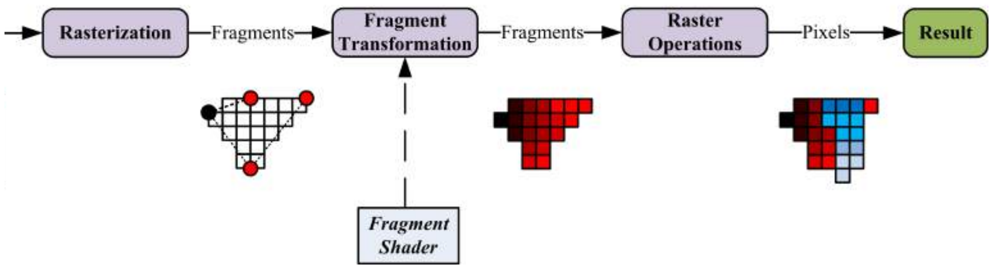
**Figure 2.7 (continued on facing page):** Stages of the GPU's graphics rendering pipeline. The vertex and fragment transformation stages are programmable, allowing creative use of the GPU for general-purpose computations.

The vertex transformation stage processes each vertex individually. At the very least, it projects the vertex from its world space position to clip space coordinates inside (or outside) the view frustrum. It can also generate or modify vertex property information. The processing is one-to-one, meaning that each input vertex must map to a unique output vertex.

The primitive assembly stage connects vertices into geometric primitives, based on the provided assembly information. The result is an ordered sequence of primitives that only move on to the rasterization stage if they pass a clipping (i.e. inside the view frustrum) and culling (i.e. front- or back-facing) test. Primitives that cross the boundary of the view frustrum are split and the outside part is discarded. To map to the requested output resolution and area, the remaining vertices are transformed from clip space to screen space via the perspective division and the viewport transform.

The rasterizer subdivides each remaining primitive into pixel-sized fragments for each pixel that the primitive covers. Different from a pixel, a fragment should be seen as a state that will eventually contribute to the final value of a pixel in the output framebuffer. It not only has a position on the screen, but also a depth value and an associated set of interpolated parameters. To determine the latter, the vertex properties are interpolated to assign a smooth gradient of values across the fragments. There is no meaningful relation between the number of vertices a primitive consists of and the number of fragments that are generated when it is rasterized. A triangle composed of just three vertices could project over the entire screen and thereby generate millions of fragments.

In the fragment transformation stage, a final color for each fragment is determined through a sequence of per-fragment operations on the rasterized properties (texture coordinates, color values, lighting components, surface normals, etc.). This stage also has the final word on the fragment's depth value and may even discard a fragment altogether. For example, if each vertex of the triangle in Figure 2.7 was assigned a color value at input, those values will at this point have been blended across the fragmented surface.



In the raster operations stage, the fragment undergoes a last series of tests. Most importantly, the depth value of the fragment is compared with the depth values of other fragments that projected to the same location and only the nearest one is retained. The stencil test can additionally be applied to restrict the area of rendering. If the fragment passes both tests, it is written to (or blended into) the framebuffer, where it finally graduates to be a pixel in the output image.

All this functionality is generally fixed in hardware, except for the vertex and fragment processing stages. Those stages can be reprogrammed by implementing a vertex or fragment shader: pieces of code with strictly defined input and output requirements that are executed for each incoming vertex and fragment separately. Meanwhile, additional GPU capabilities, such as projective coordinate generation and depth testing, are directly employable and manipulable. It is this flexible programmability that allows the creative use for purposes other than graphics rendering. The whole pipeline maps exceptionally well to the algorithmic structure of plane sweeping, as we will discover in chapter 7.

A shader is not a stand-alone application, but rather a set of strings that is passed for compilation to the GPU hardware driver from within a host application that uses a graphics rendering API. Many languages facilitate shader programming, among them most notably the industry standard GLSL (OpenGL Shading Language) [Rost et al., 2009] and Microsoft's proprietary HLSL (High-Level Shading Language) for use with Direct3D [Blythe, 2006]. We will use NVIDIA's Cg (C for Graphics) [Mark et al., 2003], of which the compiler is able to output both DirectX and OpenGL shader programs.

### 2.3.2 Compute Unified Device Architecture (CUDA)

CUDA (Compute Unified Device Architecture) is a parallel computing platform developed by NVIDIA [Sanders and Kandrot, 2010; NVIDIA Corporation, 2007]. The platform discerns two separate models: the execution and the architectural model. The execution model exposes the GPU to the programmer as a massive pool of directly accessible parallel threads. It prescribes how the algorithm should be written to be suitable for execution, what parameters need to be set and what limits need to be taken into account. The model shields the

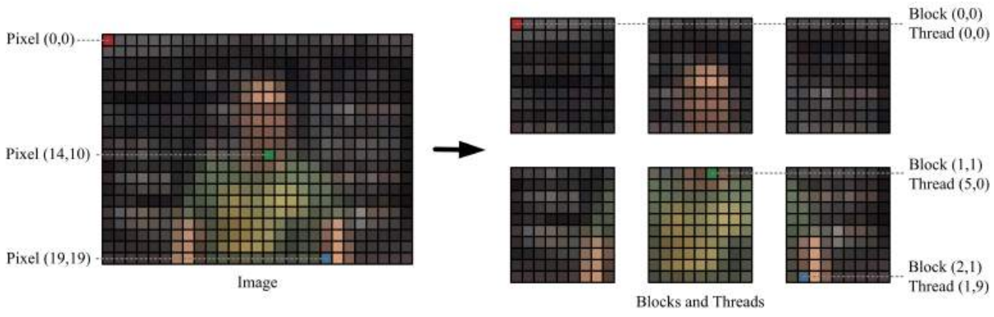


**Figure 2.8:** The CUDA executional model presents the GPU as a hierarchy of (equally sized grids of) equally sized blocks of parallel threads. The programmer can set the block size so that a one-to-one mapping between a thread and a pixel of an image emerges.

programmer of how his program will eventually be executed on the architectural level, including the actual level of parallelism. The functionality of a thread is directly specified by the programmer in a kernel which is essentially a piece of C-language code with CUDA syntax extensions.

According to the execution model, the threads are arranged in a three-dimensional hierarchy. That is to say, a (conceptual) volume of threads exists, grouped into equally sized blocks, grouped into equally sized grids. Each thread is able to uniquely identify itself inside the hierarchy by means of a series of coordinates that are retrievable at run-time. They are: the thread index $(t_x, t_y, t_z)$ (position inside the block), the block size $(B_x, B_y, B_z)$ (number of threads in the block), the block index $(b_x, b_y, b_z)$ (position inside the grid) and the grid size $(G_x, G_y, G_z)$ (number of blocks in the grid).

It is up to the programmer to set the size parameters according to the needs of his application. For example, a two-dimensional grid of two-dimensional blocks defines a layout that is ideally suited for pixel-wise operations in most image processing tasks. Given this arrangement, each thread maps exactly to one pixel coordinate $(x, y)$:

$$x = b_x \times B_x + t_x \tag{2.2}$$

$$y = b_y \times B_y + t_y \tag{2.3}$$

which is illustrated for a simplified $27 \times 20$ resolution image in Figure 2.8.

A memory hierarchy also exists. Each thread has to its disposal a number of very rapidly accessible registers for local calculations. Additionally, threads of the same block share a section of memory with equally fast access speed. This shared memory can be used for inter-thread communication and to reduce memory operations by caching data manually. To avoid dirty memory access while communicating, threads in the same block can be synchronized by the programmer. Lastly, there is global memory. Global memory is randomly accessible by all threads and by the host CPU. It is very large, but also (relatively) very slow, and accessing it should be avoided as much as possible. Global memory can also be mapped to texture memory to provide interoperability with OpenGL and other graphics rendering libraries.

On the architectural level, the dedicated functional units (e.g. the vertex and fragment shaders of section 2.3.1) have been replaced by a homogeneous collection of universal scalar floating point processors, called stream processors or CUDA cores. Each stream processor functionally represents one thread, but can also be allocated to emulate the programmable graphics pipeline. Stream processors are grouped into independent multiprocessors, with thread blocks being scheduled among multiprocessors for execution. While being executed, the threads of a block are in turn divided into groups of threads called warps. As each warp has its own program counter, the threads in a warp are physically executed in parallel. This also implies that divergent branching will not result in lost processing power, as long as all threads in a warp follow the same execution path.

The execution parameters and architectural hardware limits give rise to a number of features and concepts that the programmer is best familiar with, should he wish to successfully optimize his application. This includes memory coalescing (threads from the same half-warp can read sequential 32-bit memory addresses in one instruction), thread synchronization (warps can be suspended while being forced to wait for other warps in the block to catch up), memory latency hiding (blocks can be suspended while waiting for memory transfers) and occupancy (the ratio between actual warps being executed on a multiprocessor and the maximum possible number of warps). Discussing all the intricacies would lead us too deep down the rabbit hole; they are well documented elsewhere [Goorts et al., 2010; NVIDIA Corporation, 2015].

NVIDIA connects a compute capability to its GPUs, i.e. a version number that collects many of the execution parameters and architectural limits. Hardware is guaranteed to support any compute capability up to its own compute capability. For example, the very recent GeForce GTX TITAN Black (Kepler architecture) [NVIDIA Corporation, 2012] has compute capability 3.5, with 15 multiprocessors and 192 CUDA cores per multiprocessor (2880 CUDA cores in total), 1024 threads per block, a warp size of 32 threads, 255 registers per thread, 48 KB of shared memory per block and 6 GB of GDDR5 global memory.

Essentially, only the execution model must be understood to design a parallel algorithm. Cleverly mapping the execution model on the architectural model and designing with a specific compute capability in mind, however, will yield high levels of scalability and parallelism in the application. Considering memory coalescing, for example, it pays to create thread blocks with the width of a scanline for scanline-wise rectified stereo matching in chapter 5.

CUDA is only available for NVIDIA devices. Other parallel computing technologies that provide comparable functionality include Microsoft's DirectCompute [Boyd, 2008] and the cross-platform OpenCL [Stone et al., 2010].

## 2.4 Camera Calibration

In this dissertation, unless stated otherwise, it may be assumed that all input cameras and images have been geometrically calibrated (section 2.4.1), photometrically calibrated (section 2.4.2), corrected for lens distortion (section 2.4.3) and Bayer demosaiced (section 2.4.4).

### 2.4.1 Geometric Calibration

Consider the projection of points in space onto a plane. This is illustrated in Figure 2.9(left), with the center of projection $C$ at the origin of a euclidean coordinate frame. The plane $Z = f$ is called the image plane or focal plane, with $f$ the focal length. The ray from the camera center $C$ perpendicular to the image plane is called the principal axis of the camera and the point where the principal axis meets the image plane is called the principal point $p$. In what follows we borrow from Hartley and Zisserman [2004].

**The Pinhole Camera Model** From geometry of similar triangles in Figure 2.9(right), it can easily be derived that a point $\mathtt{X} = [X,Y,Z]^T$ in 3D world space is mapped to the point $\mathtt{x} = [fX/Z, fY/Z, f]^T$ on the image plane, where the ray joining the point $\mathtt{X}$ to the center of projection $C$ meets the image plane. By introducing homogeneous coordinates, this can be expressed as a linear mapping:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

which is known as the pinhole camera model for central projection.

**The Camera Projection Matrix** For a general camera with arbitrary position and orientation in a projective coordinate system, Equation 2.4 generalizes to:

$$\mathtt{x} \propto P\mathtt{X} \tag{2.5}$$

with $P$ the $3 \times 4$ homogeneous camera projection matrix, and $\mathtt{X} = [X,Y,Z,1]^T$ and $\mathtt{x} = [x,y,1]^T$ the homogeneous world and image coordinates respectively. Since the camera matrix $P$ maps elements of two projective spaces, it too can be regarded as a projective element with only 11 degrees of freedom. Multiplication by any non-zero scalar results in an equivalent camera matrix. The symbol $\propto$ therefore indicates equality up to a scale factor.

In the metric case, the matrix $P$ factorizes into two independent components:

$$P = K[R|t] \tag{2.6}$$

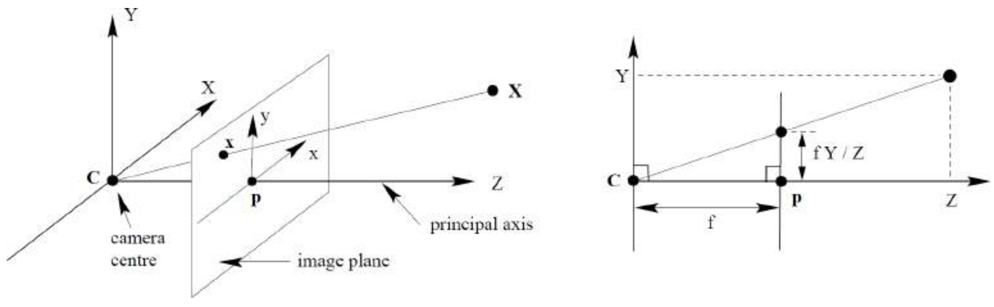


**Figure 2.9:** Pinhole camera geometry. (left) $C$ is the camera center and $p$ is the principal point. The camera center is here placed at the origin. (right) From geometry of similar triangles, the projection of a point $\mathtt{X}$ in 3D world space onto $\mathtt{x}$ on the image plane can be derived. [Hartley and Zisserman, 2004]

**Extrinsic Camera Parameters** The matrix $[R|t] = R[I| - C]$ contains the external or extrinsic camera parameters. This rigid transformation encodes the position of the camera center $C$ and the camera's orientation matrix $R$ in world space. Using this information, the coordinates of a world space point $\mathtt{X}$ within the camera's own coordinate frame (eye space) can be computed:

$$\mathtt{X}_{eye} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \mathtt{X} \tag{2.7}$$

Intuitively, we're translating the camera center back to the origin and realigning its principal axis with the $Z$ axis, as was the case in Figure 2.9(left) to begin with.

The problem of estimating the parameters of the rotation matrix $R$ and the translation vector $t$ is referred to as extrinsic camera calibration. In a setup with multiple cameras, the cameras' extrinsic calibration determines the cameras' relative position and orientation in a common world coordinate frame.

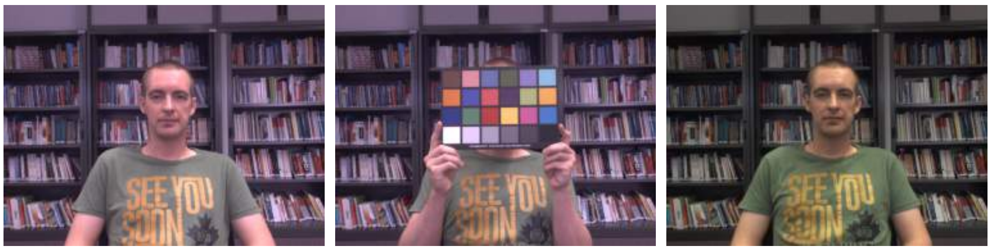
**Intrinsic Camera Parameters** Once the coordinates of the point $\mathtt{X}_{eye}$ are known, the point can finally be projected onto the image plane:

$$\mathtt{x} = K\mathtt{X}_{eye} \tag{2.8}$$

where $K$ is the intrinsic calibration matrix. It is of the general form:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{2.9}$$

with:



(a) Before: too much red. (b) During: MacBeth chart. (c) After: colors just right.

**Figure 2.10:** Photometrically calibrating the cameras will increase the reliability of pixel matches.

- $\alpha_x$ and $\alpha_y$ scale factors in the X-coordinate and Y-coordinate direction,
- $(x_0, y_0)$ the coordinates of the principal point on the image plane,
- and $s$ the skew, which is zero for most common cameras.

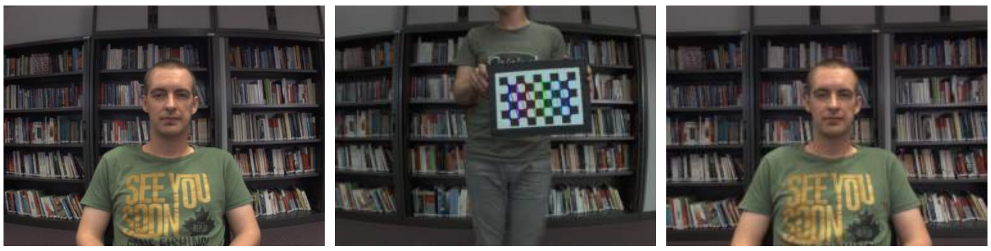
The problem of estimating these parameters is referred to as intrinsic camera calibration. When dealing with a single camera, determining the camera's intrinsic calibration often suffices.

**Calibration and Recalibration** Many libraries and toolboxes for geometric camera calibration are available. We use the one from Svoboda et al. [2005], which concurrently estimates intrinsic and extrinsic parameters from point correspondences in synchronized frames. The point correspondences are determined by waving a bright spot (e.g. a laser pointer) through the working volume.

If a camera's extrinsic calibration is lost, but its intrinsic calibration is maintained, its extrinsic parameters can be efficiently estimated again by the recalibration algorithm that we will develop in chapter 3.

## 2.4.2 Photometric Calibration

A majority of the view synthesis algorithms in this dissertation rely on matching pixels between different cameras. Hence, photometrically calibrating the cameras will increase the reliability of the matches. Photometric calibration attempts to bring the colors that the cameras capture as close as possible to the Macbeth color chart (shown in Figure 2.10(b)) by adjusting the camera's white balance accordingly. The Macbeth color chart colors were chosen to represent various natural objects, colors that are problematic for color reproduction, additive and subtractive primaries, and a gray scale [McCamy et al., 1976].



(a) Distortion not corrected. (b) Estimating parameters. (c) Distortion corrected.

**Figure 2.11:** (a) The lines of the bookshelves are clearly bent, (c) but are straightened out after the image has been corrected for lens distortion. (b) Estimating the distortion parameters involves detecting lines on a checkerboard calibration pattern.

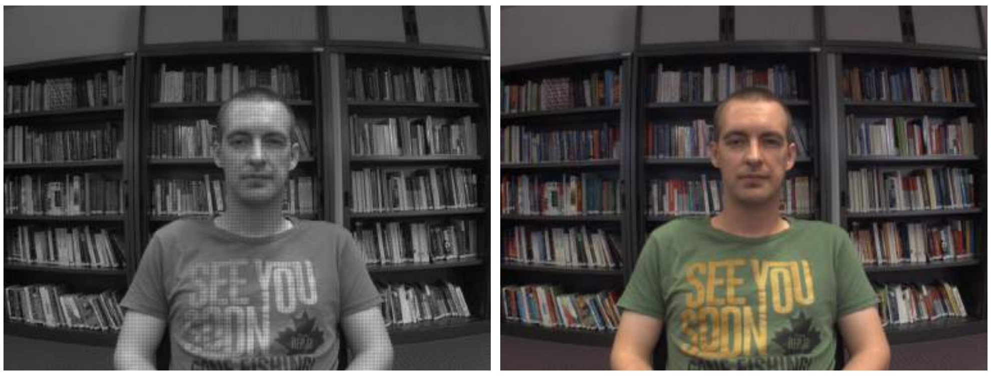
### 2.4.3 Lens Distortion Correction

We often use lenses with a short focal length to guarantee a sufficiently wide field of view. However, such lenses introduce significant non-linear distortion that cannot be modeled by the pinhole camera, because the world space point $\mathtt{X}$, its distorted projected image point $\mathtt{x}$, and the camera center $C$ are no longer collinear. This is corrected by applying the Brown-Conrady distortion model [Brown, 1966], which models both radial and tangential distortion:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = L(r) \begin{bmatrix} x_u - x_c \\ y_u - y_c \end{bmatrix} + \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} \tag{2.10}$$

with:

- $(x_d, y_d)$ the distorted pixel position,
- $(x_u, y_u)$ the undistorted pixel position,
- $(x_c, y_c)$ = the center of distortion,
- $r = \sqrt{(x_u - x_c)^2 + (y_u - y_c)^2}$ the radial distance from the center of distortion,
- $L(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \ldots$ the radial distortion function, with $\kappa_i$ the radial distortion parameters,
- $\Delta_x = (P_1(r^2 + 2x_u^2) + 2P_2xy)(1 + P_3r^2 + P_4r^4 + \ldots)$ the tangential distortion function in the X-coordinate direction, with $P_i$ the tangential distortion parameters,
- $\Delta_y = (2P_1xy + P_2(r^2 + 2y_u^2))(1 + P_3r^2 + P_4r^4 + \ldots)$ the tangential distortion function in the Y-coordinate direction, with $P_i$ the same tangential distortion parameters,



(a) Bayer pattern. (b) Demosaiced RGB image.

**Figure 2.12:** (a) A one-channel RGGB Bayer pattern must be (b) demosaiced into an RGB image.

and where ... indicates a Taylor series expansion. Usually a second order $(r^2)$ approximation suffices to achieve an accuracy of about 0.1 pixels in image space for lenses exhibiting large distortion [Devernay and Faugeras, 2001].

Many libraries and toolboxes exist to estimate the distortion parameters. We use the GML toolbox [Velizhev, 2005], which involves detecting lines on a checkerboard calibration pattern. Figure 2.11 contains an example.

### 2.4.4 Bayer Demosaicing

Typical cameras contain a CCD array of sensors that only measure light intensity. A wavelength filter is placed in front of the sensor array and only lets the desired red, green or blue wavelength of the light spectrum pass. The captured one-channel image is called a Bayer pattern, where every pixel only has a specific intensity value for its associated color wavelength. The values of the other color channels must be reconstructed from its neighboring pixels, a process that is called demosaicing and that is illustrated in Figure 2.12.

Many demosaicing algorithms have been developed, ranging from straightforward bilinear interpolation of the surrounding pixels [Ramanath et al., 2002] to gradient-based methods [Malvar et al., 2004; Goorts et al., 2012b] and advanced anti-aliasing approaches Hirakawa and Parks [2005]. An in-depth discussion is beyond the scope of this dissertation.

For our purposes, it is important to understand that the Bayer pattern is a one-channel image and thus only one third the size of its three-channel RGB counterpart. In other words, only one third of the memory bandwidth is required to transfer a Bayer pattern to the GPU for further processing. This means a considerable reduction in bandwidth consumption, which can greatly increase the overall performance in real-time applications. We will return to this when developing an end-to-end system prototype in chapter 7.

# Chapter 3

# Maintaining Camera Calibration

## Contents

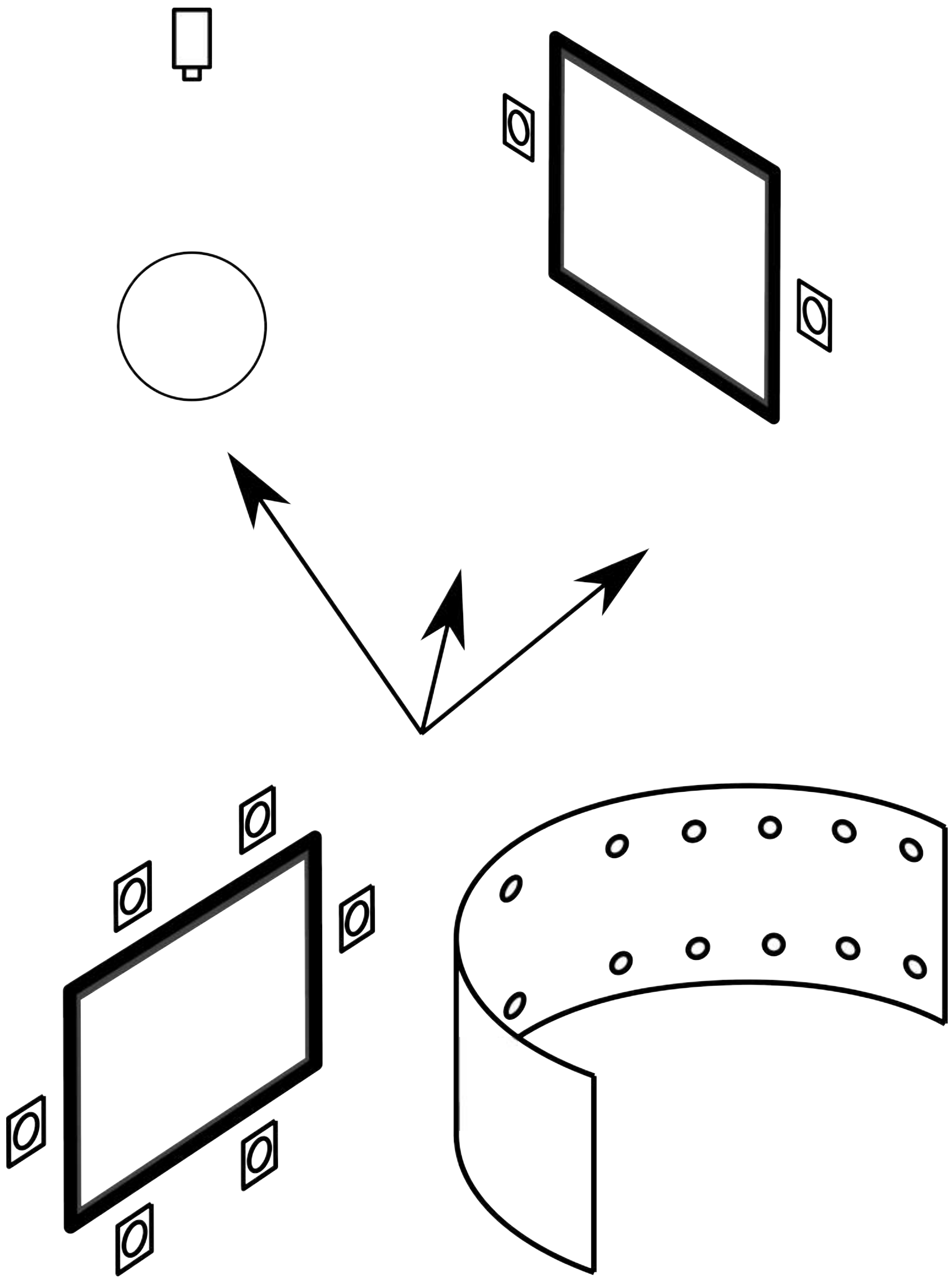


**Figure 3.1:** This chapter shows how to maintain the calibration of all the cameras depicted in the overview in Figure 1.1, p. 2.

When recording images for real-time applications, a robust camera calibration needs to be assured during the entire session. At every occurrence of camera movement, whether intended or not, it is often not feasible to interrupt the capturing to perform a full system recalibration. Even if we can shut down the system and completely recalibrate the entire camera network, this proves to be a time consuming and burdening operation. Moreover, in case of a single displaced camera, there is no immediate need to recalibrate the entire camera network. Compared with the initial calibration, there is only a small subset of changed variables, namely the extrinsic parameters of the moved camera.

This chapter presents an efficient algorithm to detect when the extrinsic parameters of a camera are no longer valid and subsequently reintegrate the displaced camera into the initially calibrated camera network by robustly recomputing its extrinsic calibration. We use available information of the remaining calibrated cameras and the known intrinsic calibration of the moved camera to reintegrate the camera in the common coordinate frame. Alternatively, when the intrinsic parameters of all cameras involved are known, the algorithm can also be applied to build ad-hoc distributed camera networks by starting from three calibrated cameras and gradually adding new cameras.

Based on image point correspondences, we compute pairs of essential matrices from the displaced camera to its neighboring cameras, which provide us with local coordinate estimates for each camera pair. These canonical pairs are related to the real world coordinates, up to a similarity transformation. From these estimates, a mean rotation and translation can be deduced in the coordinate frame of the previous full system calibration. Unlike other approaches, we do not explicitly compute any 3D structure [Hermans et al., 2007b; Goorts et al., 2014c].

This chapter shows how to maintain the calibration of all the cameras of all our prototypes in the overview in Figure 1.1, p. 2, highlighted again in Figure 3.1. It is organized as follows. Related work is first discussed in section 3.1, together with one alternative approach that is closely related to our own and therefore worth taking a closer look at. Next, working constraints that we set on the host system are described in section 3.2. The following two sections compose our recalibration algorithm. Before we should recalibrate a camera, we need to detect in section 3.3 that it has indeed moved in the first place. Once detected to have moved, the camera is then reinserted into the calibrated network of cameras by computing its new calibration in section 3.4, where our main contribution in this chapter can be found. Results are presented in section 3.5 and we conclude in section 3.6.

## 3.1 Related Work

Calibration algorithms for multi-camera setups are generally divided in four steps [Hartley and Zisserman, 2004]: (1) detect feature points in each camera and determine correspondences, (2) perform an initial reconstruction, (3) refine by applying bundle adjustment and (4) upgrade to a metric reconstruction.

In the first step, detecting and matching feature points is usually facilitated by the use of a calibration object. This object can come in many different flavors, usually either a planar surface with an imprinted binary pattern (such as a checkerboard, parallel lines [Baker and Aloimonos, 2003] or evenly spaced circles or boxes [Baker and Aloimonos, 2003; Tsai, 1987; Zhang, 2000]), or a single moving bright spot (such as a laser pointer [Svoboda et al., 2002, 2005]). All these methods provide either point-to-point, line-to-line constraints, or both, of which the accuracy is critical to the success of the rest of the algorithm.

Our algorithm will be executed during capturing time and therefore does not have the luxury of being able to use a specifically designed calibration object. Although in a controlled environment the calibration object could be an explicit part of the scene (e.g. a checkerboard pattern on the floor), we make no such assumptions. Furthermore, as our technique is intended for a wide variety of applications, we make no assumptions about the nature of the camera baselines. We employ SIFT (scale-invariant feature transform) [Lowe, 2004], the current state-of-the-art feature detector and descriptor to match over wide baselines. More recently, an interesting alternative for real-time applications has emerged in the form of SURF (speeded up robust features) [Bay et al., 2008].

The second step (the initial reconstruction) is where the key difference between most of the multi-camera algorithms lies. These techniques can roughly be labeled as either top-down or bottom-up methods.

Algorithms that employ a top-down approach are generally employed in controlled environments, such as a lab or a recording studio, calibrating the cameras all at once. A widely used approach of this category is the projective factorization proposed by Martinec and Pajdla [2002]. This method requires the estimation of projective depths for proper initialization, which is achieved using epipolar geometry as done by Sturm and Triggs [1996]. Occlusion handling is solved by an extension of the method by Jacobs [1997] to fill in missing data. This extension can exploit the geometry of the perspective camera so that both points with known and unknown projective depths are used.

Algorithms of the bottom-up category are often more suitable for less controlled environments, distributed applications [Mantzel et al., 2004], and video sequences [Fitzgibbon and Zisserman, 1998] (i.e. a dynamic scene viewed by a single camera, as opposed to multiple viewpoints of the same static scene). The general strategy here is to first perform a local calibration for one or more small clusters of cameras and then gradually converge to a solution using the previously computed building blocks.

The method used by Sinha et al. [2004] shows some resemblance to our own. They first resolve for a set of three cameras with non-collinear centers, for which the three fundamental matrices $F_{12}, F_{13}, F_{23}$ have been computed. Given these, they compute a corresponding triplet of camera matrices $P_1, P_2, P_3$. This provides a general projective frame for the rest of the reconstruction. To complete the N-view camera network, they then inductively add each of the remaining cameras. The key observation in this method is that, given camera matrices $P_1, P_2$ and fundamental matrices $F_{12}, F_{13}$, the third camera matrix $P_3$ spans a 4D subspace of $\mathbb{P}^8$.

This can be solved linearly, calculating a $\overline{F}_{23} = [e_{32}]_x P_3 P_2^+$ that most closely approximates the given $F_{23}$. The similarity with our work is that we also use information from neighboring camera matrices and fundamental (essential) matrices to restore the missing camera parameters. However, the differences between this method and ours are twofold. First and foremost, we assume the intrinsic parameters of our cameras to be known. Therefore, we are dealing with essential matrices instead of fundamental matrices. Second, unlike Sinha et al. [2004], our building block does not need to be a camera triplet. We can use information from two or more neighbors to find a solution.

The third step (refinement) consists of applying bundle adjustment to the previously computed results, usually by minimizing the reprojection error using the Levenberg-Marquardt algorithm [Moré, 1978]. This procedure results in a projective reconstruction $\{P^i, X_j\}$ of the detected feature points $\{x_j^i\}$.

If a metric calibration is desired (step four), a rectifying homography $H$ can be computed from auto-calibration constraints to upgrade the reconstruction to a metric one $\{P^i H, H^{-1} X_j\}$. To obtain this homography, we can choose between direct and stratified methods [Hartley and Zisserman, 2004]. The direct auto-calibration methods involve computing the absolute conic or its image, whilst the stratified methods solve the reconstruction in two steps: first solving for the plane at infinity, then using this to solve for the absolute conic.

## 3.1.1 An Alternative Approach

Besides the work referenced in section 3.1, one alternative algorithm stems from the following theorem.

**Theorem 3.1.** *[Hartley and Zisserman, 2004, p. 385] Given three compatible fundamental matrices $F_{21}, F_{31}$ and $F_{32}$ satisfying the non-collinearity condition, the three corresponding camera matrices $P$, $P'$ and $P''$ are unique up to the choice of a 3D projective coordinate frame.*

The first two camera matrices $P$ and $P'$ can be determined from the fundamental matrix $F_{21}$. The third camera matrix $P''$ can then be determined in the same projective frame as follows:

1. Select a set of matching points $x_j \leftrightarrow x'_j$ in the first two images, satisfying $x'^T_j F_{21} x_j = 0$, and use triangulation to determine the corresponding 3D points $X_j$.

2. Use epipolar transfer to determine the corresponding points $x''_j$ in the third image, using the fundamental matrices $F_{31}$ and $F_{32}$.

3. Solve for the camera matrix $P''$ from the set of 3D-2D correspondences $X_j \leftrightarrow x''_j$.

This is essentially the approach of triangulation (determining the 3D points $X_j$) and localization (determining camera pose from 2D-3D correspondences, more commonly known as camera resectioning) in the work of Mantzel et al. [2004]. It should be noted that they employ the described algorithm with essential matrices rather than fundamental matrices. To ensure the validity of the computed camera matrix in step three, they also have the need to ensure the orthogonality of the rotation component. This leads to an iterative algorithm that alternates between optimization of the camera rotations and translations.

An essential issue of the described method consists of the fact that it does not use all the constraints available. We choose to take a different approach, making direct use of the available information in the form of the intrinsic camera calibration, reducing the available degrees of freedom.

Another issue of this approach consists of the cumulative error that is introduced at every stage of the algorithm. Even though the approach can be extended to multiple cameras, there is still the need at every stage to remove outliers in the feature point matches. The accuracy of the camera resectioning in the third stage is also very dependent on the success of the previous stages (3D reconstruction and point-transfer) and runs the risk of becoming inaccurate in case of a low amount of point correspondences. Our approach needs to only perform a single feature matching step, exclusively in image space. This reduces the chance of introducing cumulative error to a minimum.

## 3.2 System Assumptions

We define our algorithm for a centralized network structure. More specifically, we make the following assumptions: (1) the network is synchronized and (2) the station on which the algorithm is performed has access to the neighboring cameras of the moved camera and their synchronized images. Thus we assume the algorithm to be running on a system that has access to all the nodes containing cameras, as depicted in Figure 3.2.

The algorithm can also be applied to a distributed camera network, running an instance of the algorithm at each node, assuming that the conditions mentioned above are met.

Network delays have little to no impact on the system: as long as we have a means to ensure synchronous image pairs between two cameras, all requirements for essential matrix computation are met. The time differences between acquired image pairs is irrelevant. We end acquisition when we have enough point correspondences to commence recalibration.

## 3.3 Movement Detection

The goal of our system is to ensure that all cameras in the network remain calibrated. We therefore need to detect when a camera has moved, in other words when its known external calibration is no longer valid. This movement detection algorithm loops continuously during the entire application and can be divided into two parts, shown in Algorithm 3.1.

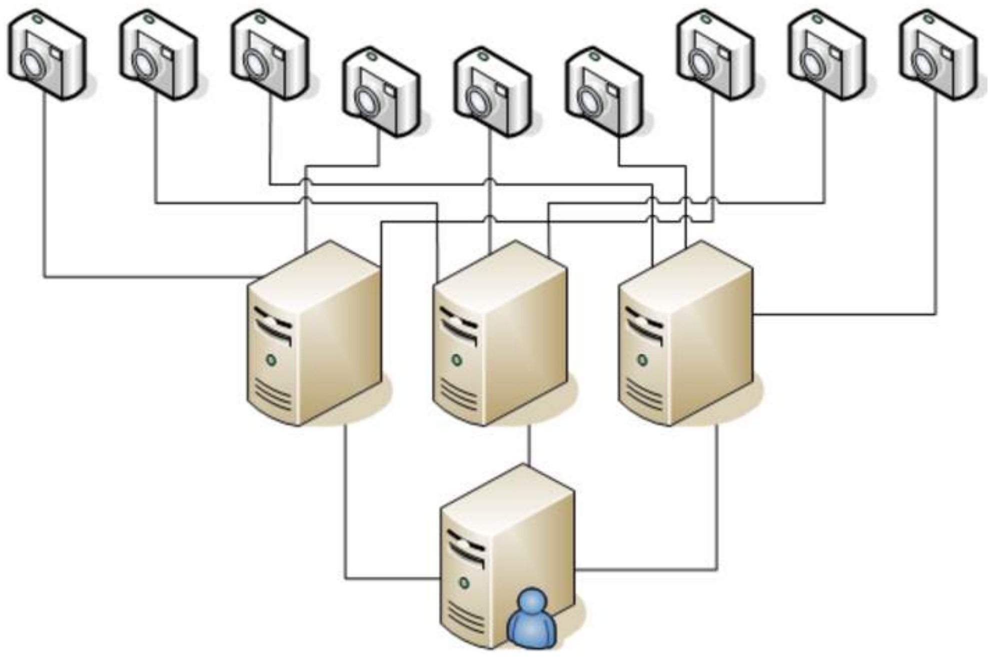


**Figure 3.2:** We define our algorithm for a centralized network structure in which the central system has access to all the nodes and their connected cameras.

### 3.3.1 Initialization

This procedure is executed at each iteration to initialize new cameras that were previously unknown to the network. When a new camera is introduced to a running application, we need to prepare it for calibration and recalibration.

First, we estimate a background from a predefined number of frames of the new camera. There are several approaches to this estimation and some are better suited for our purposes than others, e.g. we observed that applying a median filter on consecutive frames results in unnecessary sharp transitions in areas where the background is only exposed during a very limited time. This has undesirable consequences for the rest of our algorithm. Instead, we employ an averaging filter. Even so, more sophisticated approaches could be devised to improve the initial conditioning of the detection phase.

Our next step is to detect two layers of hotspots, as illustrated in Figure 3.3. We define a hotspot as an interesting feature that remains immobile during the entire recording process. This could be in any form, varying from special-purpose markers in the scene to distinctive features in the background. We opt for the latter approach, as it allows for the capturing of more general scenes.

**Algorithm 3.1** Camera Movement Detection

```
Continuous loop:

  1. Initialize new cameras since last iteration (section 3.3.1):
     (a) Compute the background image.
     (b) Determine primary and secondary hotspots.
  2. For each camera that currently is not being recalibrated (section 3.3.2):
     (a) Cross-correlate patches around primary hotspots.
     (b) If number of good primary patches lies below threshold:
          i. Cross-correlate patches around secondary hotspots.
         ii. If number of good secondary patches lies again below threshold: recalibrate
             camera (section 3.4, Algorithm 3.2).
```

As we are not interested in advanced aspects of feature detection (scale-space, affine invariance, etc. [Mikolajczyk and Schmid, 2005; Mikolajczyk et al., 2005]), but only in their *cornerness*, we have chosen to apply feature detection as done by Shi and Tomasi [1994]. The cornerness property allows us to impose a rank-order on the feature set. From the detected features, we sample two subsets which we label the primary and secondary hotspots. The primary set contains a small set of features that lie at a large spatial distance from each other, while the secondary set is larger and denser. The sets are completely independent from one another, but it is possible that a feature exists in both the primary and the secondary set. We add a predefined number of features to each set, starting with the best features (i.e. highest minor eigenvalues) and iteratively adding new features that are not too close to the features already added.

In case of a large change in the scene condition (e.g. completely different lighting), the background needs to be refreshed and the hotspots need to be recomputed. As this is a computationally expensive operation, it should only occur when, for every camera in the network, a large portion of its associated background image is different from the current frame. Alternatively, we could simply let the user trigger the background recomputation at his discretion.

### 3.3.2 Detection

For each camera that currently is not being recalibrated, we perform a double check.

First, we verify that each of the primary hotspots is still in place. For a hotspot located at pixel coordinate $(x, y)$, we perform a normalized cross-correlation between the $n \times n$ windows centered around $(x, y)$ in both the current frame and the previously computed background. If

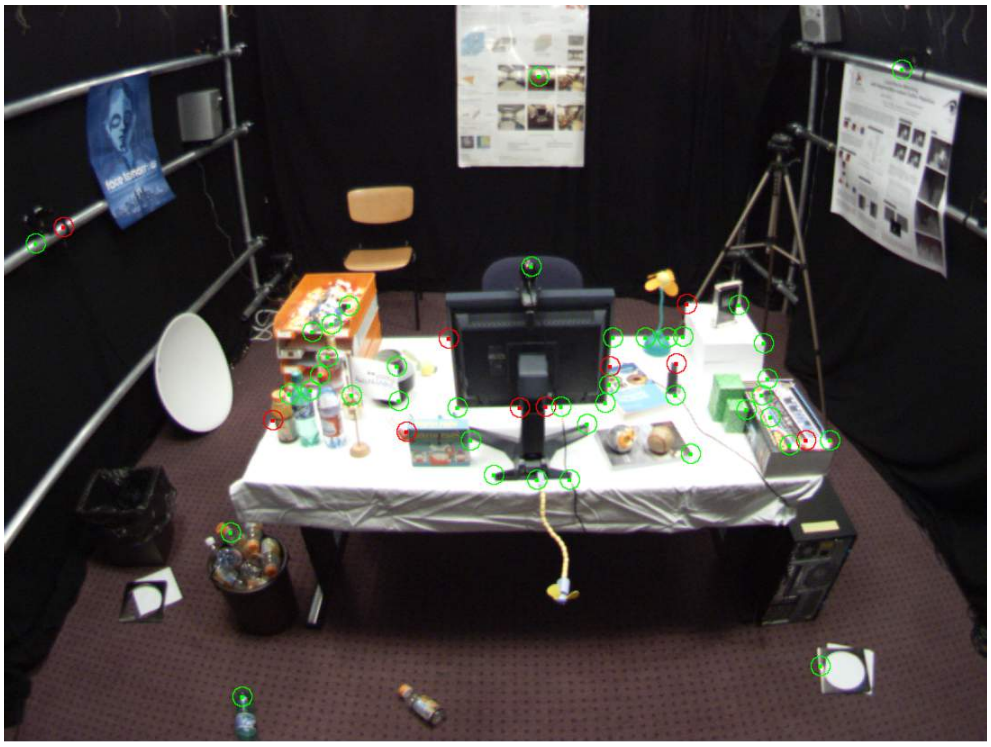


**Figure 3.3:** Given a set of detected features with an established rank-order, we can extract two subsets with high (red markers) and low (green markers) intra-element distances. We label them the primary and secondary hotspots respectively.

the computed value is below a certain threshold, this indicates that either the camera has moved or the feature point has been occluded. If a predefined number of primary hotspots for a camera can be validated, we move on to the next camera. If not, we move on to the camera's secondary layer of hotspots.

The secondary layer of hotspots serves to avoid unnecessary recalibration attempts. We repeat the algorithm described above for this additional set of feature points. If it returns a number of validated hotspots below a predefined threshold, we assume the camera needs to be recalibrated.

While our primary check is a quick process, slightly more time is needed for the secondary hotspots (approximately 0.1 to 1 ms), but this amount is still significantly less than the time-consuming SIFT feature detection and matching performed in the recalibration phase (approximately a second per camera). This approach reduces computation time to a level where the detection algorithm can run in parallel to other real-time algorithms, for which overtaxing the processor would be unacceptable.

A final note should be made for practical purposes. If the cameras are very close to the recorded scene, it is not uncommon for a user to completely occlude the background. To avoid unnecessary attempts at recalibration, it is recommended to let the camera perform an additional detection phase, a predefined amount of time after movement was first detected. This way, a passing user will not trigger the recalibration algorithm.

## 3.4 Recalibration

Once a camera has been determined to have moved by Algorithm 3.1 (section 3.3), it is flagged as inactive until the recalibration algorithm, as shown in Algorithm 3.2, is completed. The recalibration algorithm, our main contribution in this chapter, is explained from section 3.4.2 onward. But before we get there, our notation requires some clarification in section 3.4.1.

**Algorithm 3.2** Camera Recalibration

```
If Algorithm 3.1 (section 3.3) detected a camera to have moved:
  1. Assign index 0 to the moved camera $C_0$.
     Find its $N$ nearest neighbor cameras $C_i, i \in [1,N]$.
  2. For each neighboring camera $C_i$ (section 3.4.2):
     (a) Compute the essential matrix $E_{0i}$.
     (b) Compute the canonical camera matrix pair:
         $P_i^{L_i} = [I|0]$ and $P_0^{L_i} = [R_0^{L_i}|t_0^{L_i}]$.
  3. Compute the mean rotation matrix $R_0^W$ (section 3.4.4).
  4. Compute the translation vector $t_0^W$ (section 3.4.5).
```

### 3.4.1 Notation

Unless noted otherwise, we adopt the notation formalized by Hartley and Zisserman [2004]. Projective geometry and homogeneous coordinates are used.

**Normalized Cameras** Equation 2.5 and Equation 2.6, p. 26, describe the perspective projection of the scene onto the image plane. If the intrinsic calibration matrix $K$ is known, then its inverse may be applied to the point $\mathtt{x}$ to obtain:

$$\hat{\mathtt{x}} = K^{-1}\mathtt{x} = [u, v, 1]^T \tag{3.1}$$

and we refer to these coordinates as normalized coordinates. The camera matrix $K^{-1}P = [R|t]$ is called a normalized camera matrix. Going forward, we assume all our camera matrices to be of this form.

**Coordinate Frames** The Euclidean coordinate frame of the original calibration is referred to as the world coordinate frame $W$. For example, $P_1^W$ denotes the camera matrix of the first camera in the world coordinate frame.

The coordinate frames derived from essential matrices are referred to as local. For example, $P_1^{L_2}$ denotes the matrix of the first camera in the local coordinate frame of the second camera.

**Cross Product** If $a = [a_1, a_2, a_3]^T$ is a 3-vector, then its corresponding skew-symmetric matrix is defined as:

$$[a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3.2}$$

which is to say $[a]_\times$ satisfies the skew-symmetric condition $-[a]_\times = [a]_\times^T$. The cross product of two 3-vectors $a \times b$ is then related to skew-symmetric matrices according to:

$$a \times b = [a]_\times b = \left(a^T [b]_\times\right)^T \tag{3.3}$$

**Diagonal Matrix** Lastly, the notation $diag(a_1, a_2, a_3)$ denotes a diagonal matrix:

$$diag(a_1, a_2, a_3) = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix} \tag{3.4}$$

### 3.4.2 Determining Local Coordinate Frames

Let us denote the moved camera with $C_0$. The first step in Algorithm 3.2 is to find the $N$ nearest neighbors $C_i, i \in [1, N]$ of the moved camera $C_0$, with respect to its original location and orientation. If the cameras are part of nodes equipped with radio sensors and transmitters, the current nearest neighbors could be determined from broadcast intervals instead.

Once we have determined the neighbors, we compute the essential matrices $E_{0i}$ between the moved camera $C_0$ and each of its neighbors $C_i$ (step 2(a)). We employ the standard robust estimation algorithm, explained in Algorithm 3.3.

On the estimated essential matrices we apply the following theorem.

**Algorithm 3.3** Robust Essential Matrix Estimation

As described by Hartley and Zisserman [2004]:

1. **Feature point detection:** Find feature points in $N$ cameras for $M$ frames using the SIFT detector.
2. **Putative correspondences:** Determine a set of feature point matches based on the SIFT descriptor.
3. **Robust estimation:** Until a predefined threshold has been reached, use random samples of feature point matches (RANSAC, [Fischler and Bolles, 1981]) and the normalized eight-point algorithm [Hartley, 1997] to find an essential matrix $E$ with a large support of inliers.
4. **Non-linear estimation:** Re-estimate $E$ from all correspondences classified as inliers by minimizing a cost function using the Levenberg-Marquardt algorithm [Moré, 1978].
5. **Guided matching:** Further interest point correspondences are now determined using the estimated $E$.

The last two steps are iterated until the number of correspondences is stable.

**Theorem 3.2.** *[Hartley and Zisserman, 2004, p. 259] For a given essential matrix $E = U diag(1,1,0)V^T$ and first camera matrix $P = [I|0]$, there are four possible choices for the second camera matrix $P'$, namely:*

$$\begin{array}{rllll} P' = & [UWV^T|+u_3] & or & [UWV^T|-u_3] & or \\ & [UW^TV^T|+u_3] & or & [UW^TV^T|-u_3] & \end{array}$$

Testing with a single point to determine if it is in front of both cameras is sufficient to decide between the four different solutions for the camera matrix $P'$. Since these points are already available from our initial essential matrix estimation, we now have a metric local coordinate frame for each essential matrix: $P_0^{L_i} = P'$ (step 2(b)).

## 3.4.3 Linking Local And World Coordinate Frames

We will now compare the projection of a scene point $\mathtt{X} = [X\ Y\ Z\ 1]^T$ onto the image planes of any pair of cameras $(C_0, C_i)$, both in world and local coordinate frames.

The world coordinates are our reference frame, hence we dub them to be exact. This gives us the following equations:

$$\begin{array}{rcll} [u_0\ v_0\ 1]^T & = & [R_0^W|t_0^W] & \left[X^W\ Y^W\ Z^W\ 1\right]^T \\ [u_i\ v_i\ 1]^T & = & [R_i^W|t_i^W] & \left[X^W\ Y^W\ Z^W\ 1\right]^T \end{array} \qquad (3.5)$$

On the other hand, the local coordinate frame is known only up to scale. However, as we know it to be metric as well, the camera matrices are of the following form:

$$\begin{aligned} [u_0\ v_0\ 1]^T &= [\pm R_0^{L_i} | \lambda t_0^{L_i}] \quad \left[X^{L_i}\ Y^{L_i}\ Z^{L_i}\ 1\right]^T \\ [u_i\ v_i\ 1]^T &= [I|0] \quad \left[X^{L_i}\ Y^{L_i}\ Z^{L_i}\ 1\right]^T \end{aligned} \tag{3.6}$$

As the normalized image coordinates $\widehat{\mathtt{x}} = [u, v, 1]^T$ are identical in both the local and world (metric) coordinate frames, we can link these equations as:

$$\begin{aligned} \mathtt{X}^{L_i} &= \begin{bmatrix} \pm R_0^{L_i} & \lambda t_0^{L_i} \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_0^W & t_0^W \\ 0 & 1 \end{bmatrix} \mathtt{X}^W \\ \mathtt{X}^{L_i} &= \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_i^W & t_i^W \\ 0 & 1 \end{bmatrix} \mathtt{X}^W \end{aligned} \tag{3.7}$$

This consequently provides us with two equations that define the requested rigid transformation $[R_0^W | t_0^W]$:

$$R_{0i}^W = R_0^{L_i} R_i^W \tag{3.8}$$

$$t_{0i}^W = R_0^{L_i} t_i^W + \lambda t_0^{L_i} \tag{3.9}$$

where the subscript $i$ has been added to $R_0^W$ and $t_0^W$ to indicate that this is only one of $N$ possible estimates. The $\pm$ sign (direction of rotation) has also been omitted for now.

### 3.4.4 Estimating World Space Rotation

Equation 3.8 provides us with $N$ estimates of the moved camera's orientation. Using these approximate solutions, we now compute a mean rotation (step 3 in Algorithm 3.2).

Averaging rotations is not trivial and there are several approaches to this problem. Very often the barycenter of quaternions or matrices that represent the rotations are used as an estimate of the mean. One could point out that these methods neglect that rotations belong to a non-linear manifold and that to obtain proper rotations, renormalization or orthogonalization must be applied. However, Gramkow [2001] showed that using the simpler linear methods gives us a very good approximation of the non-linear average. Therefore, we employ the barycenter of rotation matrices as our method of choice.

The eigenvalues of an orthogonal matrix $R_{ort}$ are of the form $(1, e^{i\theta}, e^{-i\theta})$, where the eigenvector corresponding to the unit eigenvalue represents the rotation axis. Based on this, the mean rotation is defined as [Gramkow, 2001]:

$$\overline{R} = \arg\min_{R_{ort}} \sum_i \theta^2(R^{-1}R_i) \tag{3.10}$$

$$\approx \arg\max_{R_{ort}} tr\left(R^{-1}\sum_i R_i\right) \tag{3.11}$$

The latter optimization problem is actually the core of the 3D-3D pose problem, which has been solved by Horn [1987], Arun et al. [1987] and Umeyama [1991]. The solution can be seen as the rotation that best rotates the identity matrix into the matrix consisting of the sum of rotations. It is most easily obtained by singular value decomposition $UWV^T$, where the singular values are arranged in descending order. Introducing the matrix $S$, the mean rotation in the above problem is simply given by [Gramkow, 2001]:

$$\sum_i R_{0i}^W = UWV^T \tag{3.12}$$

$$S = diag(1, 1, det(U)det(V^T)) \tag{3.13}$$

$$\overline{R} = USV^T \tag{3.14}$$

One step has been omitted in Equation 3.12. As there are two valid possibilities for each $R_0^{W_i}$ (the omitted plus-minus sign $\pm$ in Equation 3.8), we have to designate one orientation as the dominant one. To achieve this, we first compute $R_{01}^W$. Next, we apply the following transformation for all remaining $R_{0i}^W, i \in [2,N]$, to make sure all orientations are located within the same hemisphere [Golub and Van Loan, 1996]:

$$\forall i \in [2,N] : R_{0i}^W = \arg\min_{R \in \{\pm R_{0i}^W\}} ||R_{01}^W - R||^2 \tag{3.15}$$

### 3.4.5 Estimating World Space Translation

Each local coordinate frame defines the new position of the moved camera up to scale (step 4 in Algorithm 3.2). This translates into Equation 3.9 which defines the baseline connecting the camera centers of the camera pair $(C_0, C_i)$ in the world coordinate frame. Theoretically, the moved camera $C_0$ should be located at the intersection of all these lines, as depicted in Figure 3.4. In practice, this boils down to a least-squares minimization problem in which we minimize the 3D Euclidean point-line distance. The distance $d$ between the line parametrized by Equation 3.9 and the point $\mathtt{X}_0$ is:

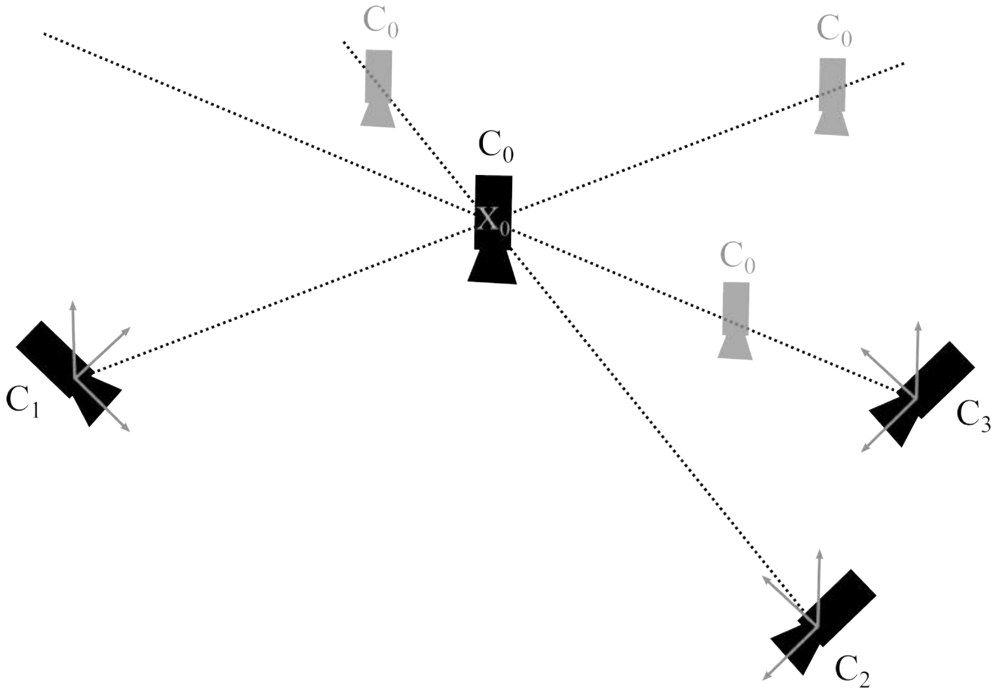


**Figure 3.4:** The local coordinate frames, obtained from essential matrices between the moved camera $C_0$ and its $N$ nearest neighboring cameras $C_i$, provide us with a baseline estimate (dotted lines) for each camera pair $(C_0, C_i)$. These estimates should have a common intersection point, reducing the problem to finding a point $\mathtt{X}_0$ with a minimized distance to each baseline. This will be the location of the new camera center.

$$d = \frac{\|t_0^{L_i} \times (R_0^{L_i}\, t_i^W - \mathtt{X}_0)\|}{\|t_0^{L_i}\|} \tag{3.16}$$

$$= \left\| \left( \frac{-[t_0^{L_i}]_\times}{\|t_0^{L_i}\|} \right) \mathtt{X}_0 - \left( -\frac{[t_0^{L_i}]_\times R_0^{L_i}\, t_i^W}{\|t_0^{L_i}\|} \right) \right\| \tag{3.17}$$

Although this definition provides us with three linear equations per neighboring camera, only two of them are linearly independent, because $[t_0^L]_\times$ is a rank two matrix (the rank of a skew-symmetric matrix is an even number).

Combining the rows of $N$ baseline estimates gives us a minimization problem of the form $\|A\mathtt{X} - b\|$, where $A$ is a $2N \times 3$ matrix and $b$ is a $2N$ vector. We find the least-squares solution to this problem using the normal equations $A^T A\mathtt{X} = A^T b$. If $A^T A$ is invertible, then the solution is $\mathtt{X} = (A^T A)^{-1} A^T b$.

## 3.5 Results

We implemented our system on a clustered network of workstations (2.8 GHz Intel Xeon processors with 2 GB RAM) connected by FireWire to four Point Grey Flea cameras at each node, counting 20 cameras in total. The recalibration algorithm was running on one of the nodes, acquiring images via network connections as needed. Camera calibration information was stored on the respective workstations.

Uniformly colored backgrounds contain little to no useful intensity variation. This is a common problem for feature matching algorithms and our movement detection is no exception to this rule. As we employ feature point detection as the basis for our hotspots, the movement detection routine depends on a sufficient amount of background texture. A possible workaround is to provide the scene with markers serving as traceable hotspots. In a dense camera network these markers could be assigned to the cameras themselves, allowing the cameras to track each other. Processing time for the detection phase is in the order of $10^{-4}$ to $10^{-3}$ seconds per iteration, making it attractive for real-time applications.

Recalibration time is severely reduced when compared with full system recalibration, even though there is still the matter of the time consuming SIFT detector. The time needed to detect and describe SIFT features in each camera frame proves to be the main bottleneck of our system (approximately a second per camera). In comparison, we also implemented the algorithm for small-baseline setups, using a normalized cross-correlation of RGB intensities as our matching criterion. This results in a significant reduction in computation time, often to less than a second. For unknown scene configurations, however, a wide-baseline setup must be assumed and SIFT is currently still the state-of-the-art descriptor available. If aiming for real-time applications, using SURF [Bay et al., 2008] should result in speed improvements comparable to the small-baseline implementation.

We have observed the recalibration algorithm to give accurate results in practice, on par with state-of-the-art (de-)centralized network recalibration algorithms. Using the newly computed camera matrix, the essential matrices associated with its neighbors produce a reprojection error of less than a pixel.

## 3.6 Conclusion

We developed a novel algorithm to robustly maintain the calibration of a camera network after the event of an (un-)intended camera displacement, without the need for a full system recalibration. The algorithm detects when the extrinsic parameters of a camera are no longer valid and reintegrates the displaced camera into the previously calibrated camera network.

Detection of displacement is done by means of hotspots, i.e. traceable features in the background image. A sparse and a dense layer of features are matched in a temporal sequence of frames, to establish whether they are still present and located at the same pixel coordinates.

If a large enough quantity of the hotspots remain in place, the camera is deemed stable. If not, the displaced camera is recalibrated.

Recalibration is achieved using image point correspondences only, without the need to compute the underlying 3D scene structure. We compute essential matrices from the displaced camera to its neighboring cameras, providing us with local coordinate estimates for each camera pair. These canonical pairs are related to the real world coordinates, up to a similarity transformation. From these estimates, a mean rotation and translation is deduced in the coordinate frame of the original full calibration.

It is important to note that our calibration algorithm is based on fitting camera pair estimates, rather than fitting 2D-3D correspondences. This means our computations have a lower dimensionality (two rotations and translations instead of a large set of point correspondences) and avoid the unneeded step of computing the 3D estimates.

The results of our algorithm are comparable to those of state-of-the-art (de-)centralized network recalibration algorithms.

### 3.6.1 Future Work

The main limitation of our algorithm is that it is unable to discern between a change in extrinsic and intrinsic camera parameters. If a change is detected, it automatically assumes that it has physically moved (i.e. its extrinsic parameters have changed), even though a change in intrinsic parameters might have occurred instead, e.g. a change in the zoom level.

# Chapter 4

# Environment Remapping

## Contents

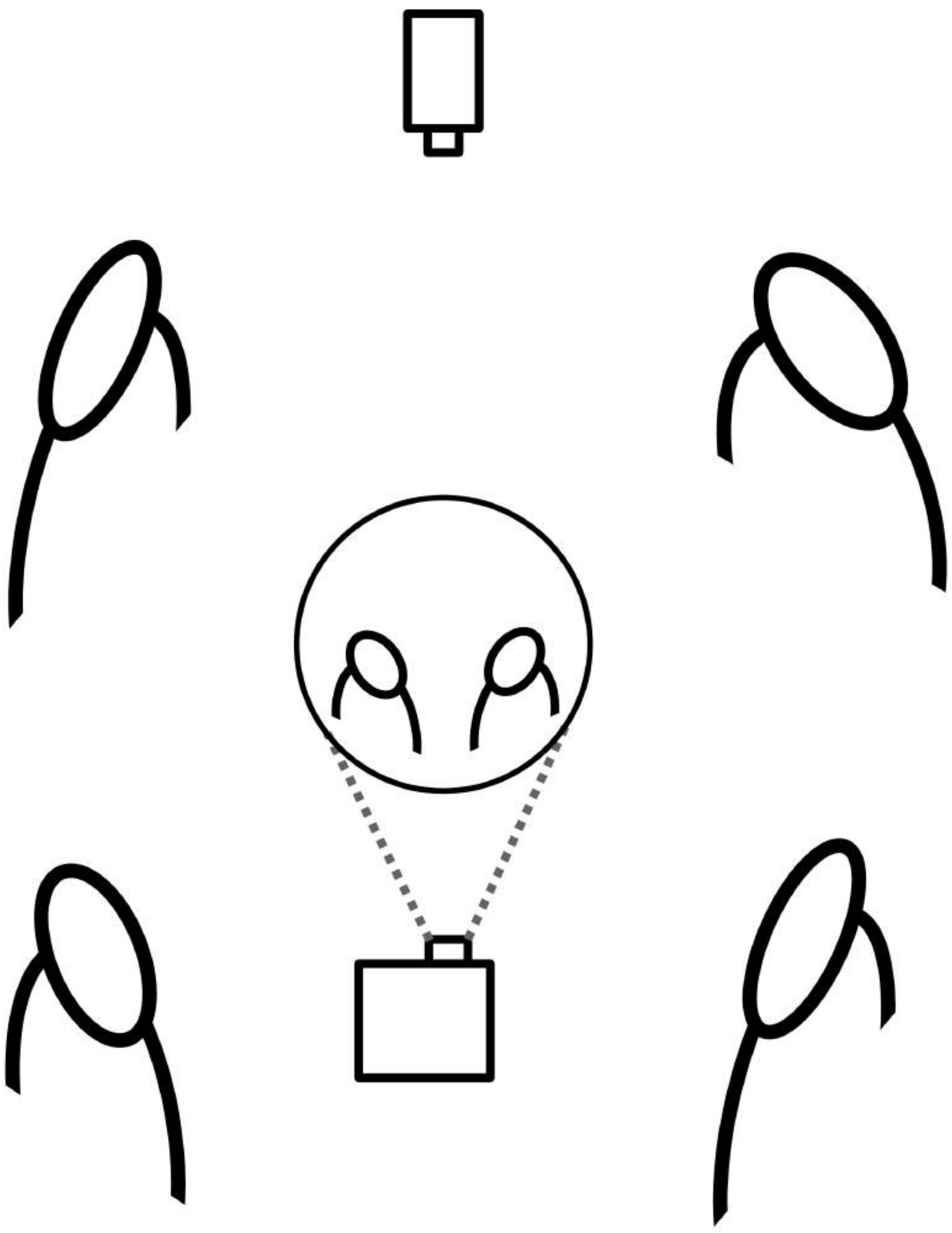


**Figure 4.1:** This chapter remaps the environment to implement the minimal one-camera solution depicted in the overview in Figure 1.1(a), p. 2.

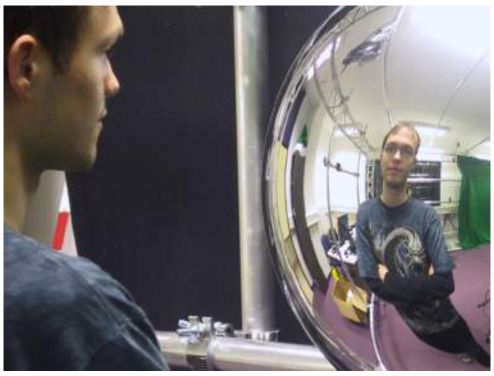
We start off the main part of this dissertation with a somewhat unconventional solution to the problems introduced in section 1.1, p. 3, that is based on the concept of environment mapping. We placed environment mapping in the context of depth-image-based rendering without geometry in section 2.1.1, p. 12.

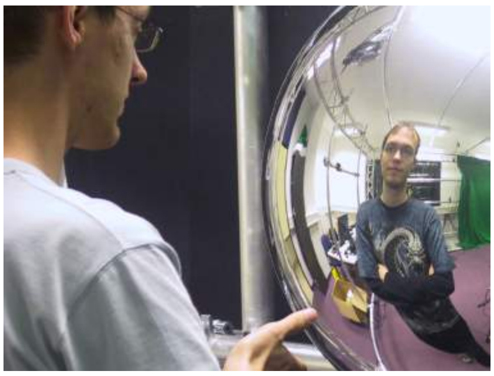
Two key observations are illustrated by Figure 4.2(a). First, when looking head-on into a spherical mirror, a user looks into his own eyes. This implies correct eye gaze, as his gaze immediately locks on to his reflection's gaze. Second, this happens without losing a clear view of the environment, exactly due to the non-planar spatial nature of the viewing window. As artificially composed in Figure 4.2(b), being able to exchange such views between two (groups of) people would create a natural way of communicating. This is the core idea behind this chapter.

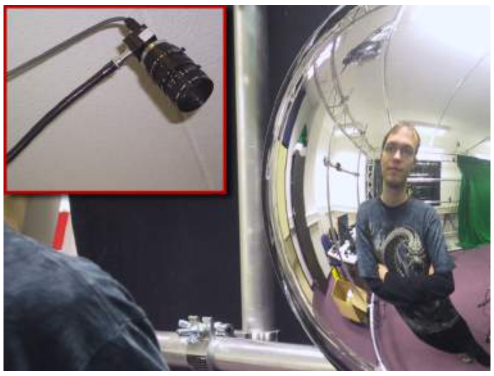
To realize this idea, we combine the capture of omnidirectional video (in other words, the environment) by filming a spherical mirror (shown in Figure 4.2(c)) with corresponding projection on a diffuse white spherical screen (represented in Figure 4.2(d)). Both capture and display are performed on a single sphere, forming a single communication device. In essence this creates a virtual overlap between the screen and the camera, which results in the user looking directly into the camera while at the same time looking at the display. We have dubbed this single device a *JanusLight*, after the Roman two-faced god, and will henceforth refer to it by this name [Hermans et al., 2006, 2007a].

The unconventional novelty in this approach is that we correct eye gaze *without* the need of view interpolation, the latter which will be the subject of the following chapters. Instead, this chapter demonstrates that the same effect can be achieved by capturing and projecting video on appropriately curved rather than planar surfaces. With a given external calibration of the camera-projector setup, the pixel-to-pixel mapping used for the image transformation has to be computed only once and our solution essentially becomes one of environment *remapping*.

This chapter corresponds to the minimal one-camera solution, depicted in the overview in Figure 1.1(a), p. 2, and enlarged in Figure 4.1. It is organized as follows. We set off by discussing related work in section 4.1. Next, we define two possible user configurations (many-to-many and one-to-one) for our JanusLights in section 4.2. Each configuration requires a different image mapping between the captured input and the projected output and we develop the mathematical equations for those mappings in section 4.3. This is followed by a presentation of various application scenarios in section 4.4. How we construct our proof-of-concept prototypes is explained in section 4.5, after which results are presented in section 4.6. In section 4.7 we discuss how our prototypes measure up to the requirements laid out in section 1.1, p. 3, and we conclude in section 4.8.


(a) Looking into a spherical mirror.


(b) (Artificially) exchanged views.


(c) Capturing omnidirectional video.

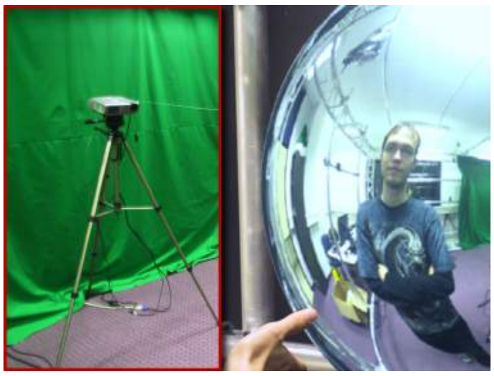

(d) Projecting on a spherical screen.

**Figure 4.2:** (a) When looking head-on into a spherical mirror, a user looks into his own eyes without losing a clear view of his environment. (b) If we'd be able to exchange such views between two people (accomplished artificially here), we'd create a natural way of communicating. (c) We realize this core idea by combining the capture of omnidirectional video from a spherical mirror, (d) with the corresponding display by projection on a diffuse white spherical screen.

## 4.1 Related Work

We introduced environment mapping and projection mapping and placed them in the context of image-based rendering in section 2.1, p. 11. They can also be placed in the context of image-based lighting (IBL) [Debevec, 2002], i.e. to simulate real-world lighting or to project images as light sources.

**Environment Mapping** An environment map is an omnidirectional representation of real-world light information, captured as an image (or texture) [Blinn and Newell, 1976]. It can be used to simulate highly detailed real-world lighting for objects in a computer-generated scene, thereby avoiding the need to accurately model or simulate illumination using a more complex rendering technique (e.g. ray tracing).

Debevec [1998] presents a method that uses measured scene radiance and global illumination in order to add virtual objects to real-world scenes with correct lighting. He computes the scene radiance by filming a specular reflective sphere in the center of the distant scene. As he demonstrates in Figure 2.3, p. 13, unfolding the environmental reflection captured on a specular sphere yields omnidirectional imagery with a projection center located at the center of the sphere. For this to be valid, he necessarily makes two assumptions: a (relative to the environment) small sphere and an orthographic (affine) camera projection. Nevertheless, this observation is key to our solution for eye gaze correction in this chapter, as we film a spherical mirror to similarly provide us with the light intensities coming from all directions and arriving at the center of the sphere.

Very recently, Michiels et al. [2014] proposed a technique to augment video with new virtual objects that are rendered under the same lighting conditions of the video. Realistic rendering is achieved by applying the omnidirectional video frames as environment lighting in the rendering equation.

**Projection Mapping** The goal of the recent field of projection mapping, also known as spatial augmented reality, is to deform an image that is to be projected in such a way that it projects in correct perspective on a real-world, often irregularly shaped, object [Bimber and Raskar, 2005]. Moving away from conventional planar displays, previously static objects can be augmented to include dynamic lighting, notions of movement, optical illusions, etc.

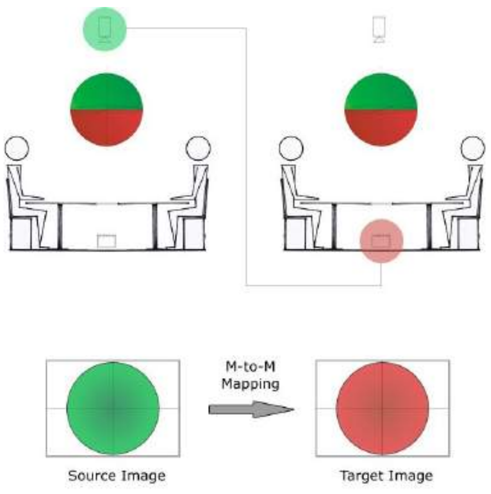
Originally conceived as shader lamps by Raskar et al. [2001], they animate neutral objects – diffuse white objects with a defined geometry, but without any texture information – with image-based illumination. In later work they extend the concept with self-configuring geometry-aware projectors [Raskar et al., 2003].

Very recent projection mapping research includes Microsoft's IllumiRoom [Jones et al., 2013] which augments the space surrounding a television screen with peripheral projected illusions to enhance traditional gaming experiences. The illusions can relight the user's living room, react to what happens in the game by distorting reality, extend the field of view, etc. A Microsoft Kinect captures the room, after which a projector projects an image that has been compensated for the specific color and geometry of the room around the television screen.

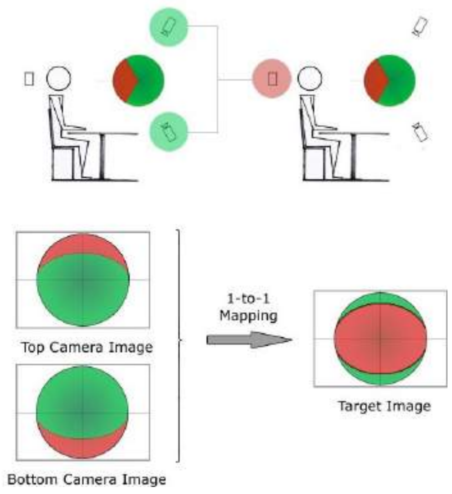
## 4.2 Configurations

Our JanusLights – partly diffuse, partly specular spheres – are used in one of two configurations:

- The **many-to-many configuration** has one camera at a distance on top, and a projector from below, while the users are located in the lower horizontal space around the



(a) The many-to-many configuration.



(b) The one-to-one configuration.

**Figure 4.3:** JanusLights can be set up in two configurations, each with their own usage scenario and image mapping. (a) The many-to-many configuration has one camera capturing from a distance at the top and a projector displaying from below, while the users are located in the 360-degree space around the sphere. (b) The one-to-one configuration has a triangular camera-projector setup with two cameras capturing from behind the sphere and a projector displaying on the front of the sphere. Green (input) indicates specular surfaces recorded by the camera(s), while red (output) represents diffuse white projection surfaces.

sphere. Capture and projection are performed by a single camera and projector. The image captured from the specular northern hemisphere is transformed by the many-to-many image mapping and the remapped image is projected on the diffuse southern hemisphere. A typical setup is depicted in Figure 4.3(a), where the users sit around a meeting table with a hole in the center.

- The **one-to-one configuration** has a triangular camera-projector setup with two cameras behind the sphere and a projector in front of the display screen. The user's viewing direction is the same as that of the projector, so the projector is put above the shoulder of the user. The images captured by the cameras are transformed according to the one-to-one image mapping and the remapped image is projected on the display surface of the JanusLight. To acquire the needed information from both cameras, we need to slightly reduce the angle of the vertical field of view for this setup. This results in an eye-like display screen, as can be seen in Figure 4.3(b).

## 4.3 Image Mapping

Similar to procedures that compute environmental reflection, we can map points on the spherical mirror to directions of reflected light on the display surface, as if the display wasn't diffuse at all, but a perfect mirror itself. As long as the gaze direction of the user meets the center of projection (which it should, as the user looks into the center of the sphere head-on), he automatically looks into the camera when watching the display and thus eye gaze is correct.

We first describe a general camera-projector mapping in section 4.3.1. From this general mapping, we specify the variables for both the many-to-many and the one-to-one configuration in section 4.3.2 and section 4.3.3 respectively. The notation refers to the vectors shown in Figure 4.4.

### 4.3.1 General Mapping

Without loss of generality, we assume the $(x,y)$ image coordinates of the input cameras to be centered on the image of the captured sphere. On the projector side, we make the similar assumption that the pixel at $(0,0)$ is projected onto the center of the diffuse surface of the sphere. In both configurations we perform a backward mapping, starting in the coordinate frame of the receiver (output) and ending in the coordinate frame of the sender (input).

#### 4.3.1.1 Output

To accommodate for possible imperfections in the constructed hardware, in actuality we model the JanusLight as a spheroid $Q_{JL}$ rather than a perfect sphere. This provides us with a first equation for any point $q = [X_q, Y_q, Z_q, 1]^T$ on the spheroid's surface ($q \in Q_{JL}$):

$$\begin{bmatrix} X_q & Y_q & Z_q & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{a^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 & 0 \\ 0 & 0 & \frac{1}{a^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} X_q \\ Y_q \\ Z_q \\ 1 \end{bmatrix} = 0 \tag{4.1}$$

The input of a JanusLight is omnidirectional video with the projection center in the center of the sphere. For our calculations this implies the use of an affine camera model. This assumption is valid in case of a substantially large camera-sphere and projector-sphere distance, relative to the radius of the sphere. The larger the JanusLight, the larger the required distances. To retain uniformity, we make a similar assumption at the output side. The affine camera matrix is defined as:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

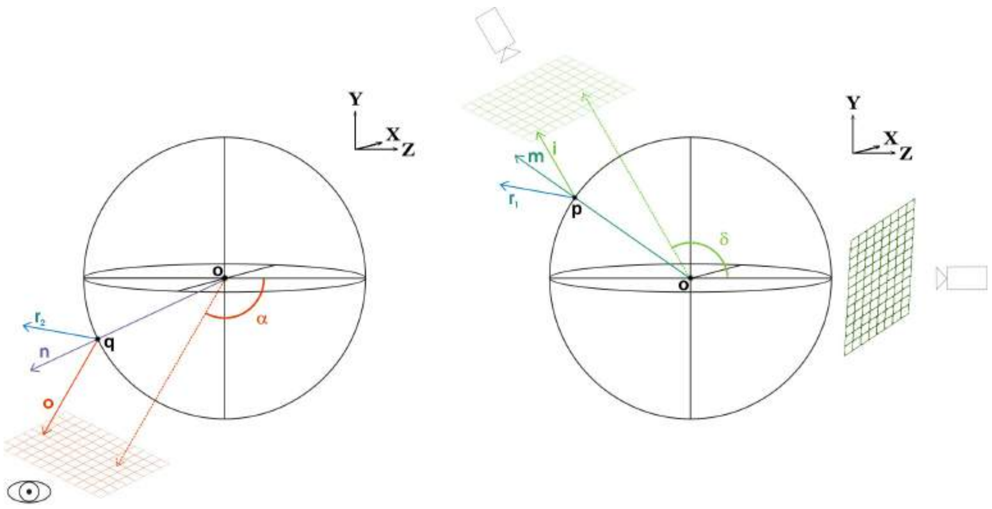


**Figure 4.4:** In both configurations, we use a backward mapping from projected pixels to camera pixels. Rays coming from the projector (direction $\vec{o}$) are mapped to their normal $\vec{n}$ or more naturally, their reflection direction $\vec{r_2}$. Using the corresponding reflection $\vec{r_1}$ on the input side and the direction of the camera $\vec{i}$, we can find the intersection of the normal $\vec{m}$ and the coordinate of the wanted pixel in the camera image.

We also know the position(s) of the camera(s), which can be reached with a rotation around the X-axis with angle $\alpha$. Combining this knowledge with the affine model gives us:

$$\begin{bmatrix} x_{out} \\ y_{out} \\ 1 \end{bmatrix} = A\ R_X(\alpha) \begin{bmatrix} X_q \\ Y_q \\ Z_q \\ 1 \end{bmatrix} \tag{4.3}$$

Putting these first three equations together, we can see that equating the light intensities arriving at point $q = [X_q, Y_q, Z_q, 1]^T$ on the JanusLight to the image coordinates $(x_{out}, y_{out})$ of the projector, results in the following group of constraints:

$$\begin{cases} X_q = x_{out} \\ Y_q = \frac{y_{out} - sin(\alpha) Z_q}{cos(\alpha)} \\ Z_q = \frac{y_{out} - cos(\alpha) Y_q}{sin(\alpha)} \geq 0 \\ \frac{X_q^2 + Z_q^2}{a^2} + \frac{Y_q^2}{b^2} = 1 \end{cases} \tag{4.4}$$

Our goal on the receiving side is to mimic the illumination of a mirror on the sending side. In order to do this, we need to compute the reflection direction for each point $q \in Q_{JL}$. We know that for each point X on a quadric Q, the tangent plane $\pi_X$ is found:

$$\pi_X = QX \tag{4.5}$$

The tangent plane $\pi_q$ in point $q$ defines the normal $\vec{n}$:

$$\vec{n} = \left[\frac{X_q}{a^2}, \frac{Y_q}{b^2}, \frac{Z_q}{a^2}, 0\right]^T \tag{4.6}$$

Given this normal and the projector (output) direction $\vec{o}$, the corresponding reflection direction $\vec{r}$ is computed as:

$$\vec{r} = 2\vec{n}(\vec{n} \cdot \vec{o}) - \vec{o} \tag{4.7}$$

#### 4.3.1.2 Input

On the sending side, we need to map from the reflection direction to the corresponding image pixel in the camera. First we compute the normal from the reflection direction and the camera direction:

$$\vec{m} = \frac{\vec{r} + \vec{i}}{\|\vec{r} + \vec{i}\|} = [X_m, Y_m, Z_m, 0]^T \tag{4.8}$$

The light ray reflecting on the JanusLight according to the given reflection direction, touches the spheroid at a point $p$. The corresponding tangent plane in $p$ is defined as:

$$\pi_p = [a^2 X_m, b^2 Y_m, a^2 Z_m, -1]^T \tag{4.9}$$

From this equation, we can derive $p$ itself:

$$\begin{gathered} p = [X_p, Y_p, Z_p, 1]^T \in \pi_p \cap Q_{JL} \\ \Downarrow \\ p = \frac{[a^2 X_m, b^2 Y_m, a^2 Z_m, \sqrt{a^2 X_m + b^2 Y_m + a^2 Z_m}]^T}{\sqrt{a^2 X_m + b^2 Y_m + a^2 Z_m}} \end{gathered} \tag{4.10}$$

Once we know the location of the surface point $p$, we can seek its projection in the corresponding camera – with angle $\theta$ around the X-axis – thus completing the transformation:

$$\begin{bmatrix} x_{in} \\ y_{in} \\ 1 \end{bmatrix} = A\, R_X(\theta) \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix} \tag{4.11}$$

### 4.3.2 Many-To-Many Mapping

In this configuration a single camera is positioned at angle $\theta = \frac{\pi}{2}$ in the YZ-plane, while the projector is located at an angle $\alpha = -\frac{\pi}{2}$. The many-to-many image mapping varies slightly from the general one. The reason for this is twofold: (a) users look at the JanusLight from the sides instead of standing right in front of it ($\alpha = -\frac{\pi}{2}$), and (b) multiple users use the JanusLight at the same time, so the projected image has to be view independent.

Our proposed approach is to map the reflection $\vec{r}$ in the receiving JanusLight to the normal $\vec{n}$ in the sending one:

$$\begin{aligned}
&(X_q = x_{out}) \wedge \left(Y_q = \sqrt{b^2 - \tfrac{b^2(x_{out}^2+y_{out}^2)}{a^2}}\right) \wedge (Z_q = y_{out}) \\
&\Downarrow \\
\vec{n} &= [X_n, Y_n, Z_n, 0]^T \\
&= \left[\frac{x_{out}}{a^2}, \frac{\sqrt{b^2 - \frac{b^2(x_{out}^2+y_{out}^2)}{a^2}}}{b^2}, \frac{y_{out}}{a^2}, 0\right]^T
\end{aligned} \tag{4.12}$$

According to the law of reflection ($\vec{r} = 2\vec{m}(\vec{m}\cdot\vec{i}) - \vec{i}$), this normal gets mapped to reflection $\vec{r}$ in the sending JanusLight:

$$\begin{aligned}
&(\vec{m}\cdot\vec{i})\vec{m} = \tfrac{\vec{r}+\vec{i}}{2} \\
&\Downarrow \\
&Z_m[X_m, Y_m, Z_m, 0] = -\tfrac{[X_n, Y_n, Z_n - 1, 0]}{2} \\
&\Downarrow \\
&Z_m \leftarrow \sqrt{\tfrac{1-Z_n}{2}} \wedge X_m \leftarrow -\tfrac{X_n}{2Z_m} \wedge Y_m \leftarrow -\tfrac{Y_n}{2Z_m}
\end{aligned} \tag{4.13}$$

Derivation of the corresponding image point from the normal on the sender's side can be done analogously to the derivation in the general mapping.

### 4.3.3 One-To-One Mapping

In this configuration, we can directly apply the general image mapping described earlier. More specifically, the cameras are positioned at angles $\alpha = 0$ and $\alpha = \frac{2\pi}{3}$ in the YZ-plane, while the user looks at the scene from $\theta = -\frac{2\pi}{3}$ degrees. Depending on the pixel position in the projected image ($y_{out} \leq 0$ or $y_{out} > 0$), the system chooses the correct camera to perform the mapping.

## 4.4 Applications

Depending on the configuration used, we can have different application scenarios. These include:

- The combination of a diffuse and a specular hemisphere, mapping reflections on the projector side to normals on the camera side, generates sensible images from any point of view around the device. This allows for *many-to-many communication*. A meeting room with a central table would be an excellent candidate for the use of our JanusLights as video conferencing devices. Figure 4.7 illustrates this application scenario.

- A triangular camera-projector setup is suited for *one-to-one communication*. Typical applications would be standard video conferencing, be it a stand-alone system like a telephone or a webcam-like extension for a personal computer. An example of this application can be seen in Figure 4.8

- JanusLights can also be used as *decorative lighting devices*, offering a point of view into (network-) linked spaces, such as living rooms, pubs, halls, theaters, events, etc. Figure 4.9 shows the many-to-many setup at work as an eye-catcher at an art exposition at a local theater.

Figure 4.5 shows an attractive artist's impression of these scenarios as imagined in a day-to-day personal living room environment.

## 4.5 Implementation

For our prototypes, we used off-the-shelf hemispherical mirrors, the kind typically used for shop surveillance. Two of these stuck together form a sphere. In the case of the many-to-many configuration, one hemispherical mirror is completely coated in diffuse white paint to project on, whereas for the one-to-one configuration only a quarter of the whole sphere is coated. Our prototype builds are shown in Figure 4.6, as the realization of the schematics of Figure 4.3.

As the mirrors are neither perfectly spherical nor specular, image quality suffers most. As explained in section 4.3, we attempt to compensate for these spatial deformations by modeling our sphere as a spheroid with a vertical axis of a different length than the horizontal ones. Another hardware limitation on image quality consists of the inherent resolution restrictions of the projector and the cameras. It is important to note that both issues are related to the hardware of the prototype rather than to the system concept.

In order to balance the perceived intensities of the pixels when projected on the spherical display, each pixel in the projected image is best multiplied by an attenuation factor that

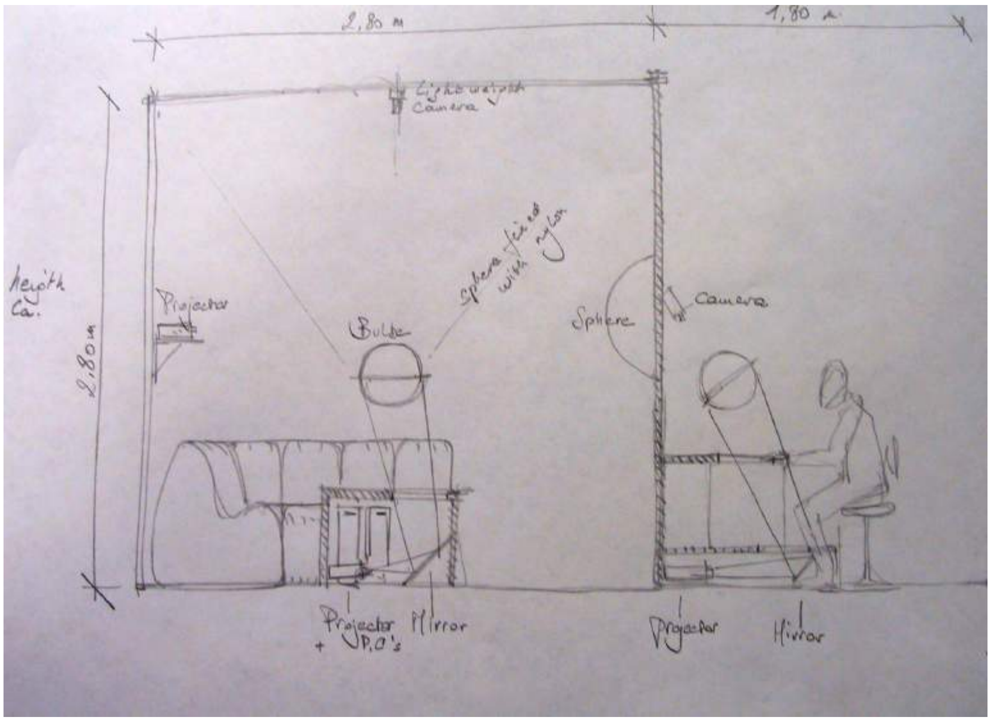


**Figure 4.5:** Artist's impression of JanusLights, used in various application scenarios and set in a day-to-day personal living room environment. Sketch courtesy of Eric Joris [CREW].

depends on the distance from the projection center. This factor results in an increased luminance value for pixels in the outer rim when compared with center pixels, noticeable in Figure 4.7(c).

All image manipulations are straightforward pixel-wise 2D or 3D operations which makes them very suitable for implementation and execution on the GPU. Moreover, most operations need to be computed only once, ahead of the rendering step. The pixel-to-pixel mapping code, for example, can be computed as a preprocessing step, which produces a reference table that can be uploaded to the GPU as a texture.

Lastly, we extend the basic system in several ways to provide the user with more options for personal use. These additions have no significant impact on system performance:

- During many-to-many communication, users can easily rotate the view by performing a simple 2D rotation on the captured scene. When used as a decorative lighting device, applying a small constant rotation results in an increased feeling of affiliation with the linked space, as a local non-moving user is able to gradually get a full mental image of the remote site.

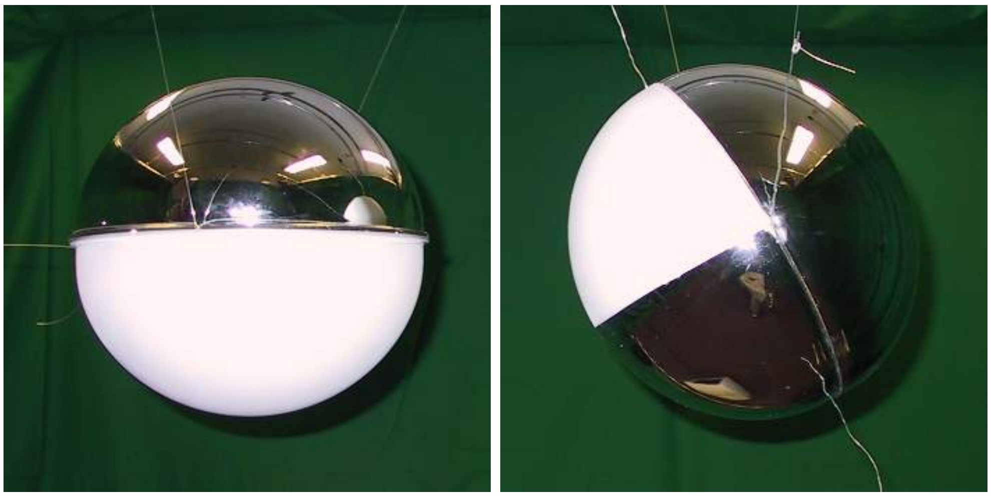

(a) The many-to-many prototype. (b) The one-to-one prototype.

**Figure 4.6:** Our prototypes are built from off-the-shelf hemispherical shop surveillance mirrors, stuck together to form a full sphere. (a) In the case of the many-to-many configuration, one hemispherical mirror is completely coated in diffuse white paint to project on. (b) For the one-to-one configuration only a quarter of the whole sphere is coated. Compare with Figure 4.3.

- The many-to-many output can be divided in a number of equally sized slots – much like a pie-chart – equal to the number of users on the viewing side. Together with the rotations mentioned in the previous extension, this gives each user an individual viewing window under his control. In the same way, these slots can easily be adapted to support multi-party communication.

## 4.6 Results

Our lightweight implementation easily operates at a real-time rendering speed of at least 25 FPS on an NVIDIA GeForce 8800 GTX graphics card (Tesla architecture) [Lindholm et al., 2008] with no mentionable delay other than network and frame grabbing delays. This also accounts for the extensions described in section 4.5.

Overall, our many-to-many (Figure 4.7) and one-to-one (Figure 4.8) prototypes look promising. Our experiments have shown that the concept works, despite technical difficulties. In the next sections we will address these difficulties and any remaining artifacts.

### 4.6.1 Many-To-Many Results

As the light rays emitted from the projector are bundled like a cone, projection on the edges of the display side is suboptimal (Figure 4.7(a)). There are two possible ways to resolve this issue. The first option is to place concave mirrors to bend the projected rays to their corresponding 3D coordinate on the sphere. This method still uses an external projector and thus requires additional careful calibration. The second option allows fixed one-time factory-calibration by placing the projector inside the sphere. Spherical display systems with the projectors stored on the inside, such as the Omniglobe [ARC Science Simulations, 2003] or the Magic Planet [Global Imagination, 2002], provide excellent alternatives, but they are more difficult to construct. Furthermore, using internal projections would obviously alter our mapping method, as the output mapping will become that of a concave mirror, yet the main principles remain the same.

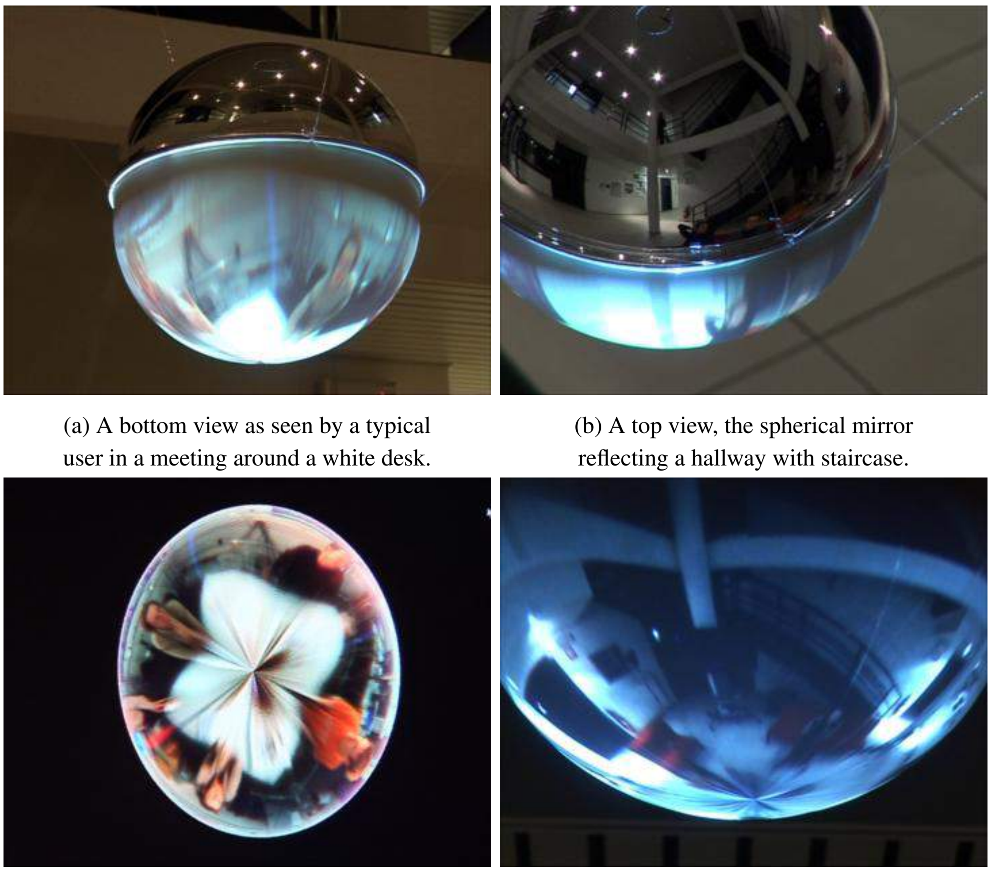
Another noticeable issue is the vanishing point at the center of the projected image. This occurs because of the poor quality of the information gathered from the edge of the captured sphere. This is understandable, as there is no way to view the content directly beneath the sphere. Fortunately for our purposes, discomfort is minimized by two facts. First, in an office scenario a JanusLight is usually placed above a uniformly colored meeting table, which means the pixels associated with the artifact are roughly all of the same color. Second, no user is positioned at the viewing angle where the artifact occurs in the first place. This is clear from Figure 4.7(c), where the vanishing point is located at the center of the white conferencing table.

Finally, Figure 4.9 shows two network-linked JanusLights in a many-to-many configuration used as decorative lighting devices at an art exposition at a local theater.

### 4.6.2 One-To-One Results

Our one-to-one setup cannot afford to suffer from the vanishing point artifact, because it would be located in the center of the display screen. Hence we opted for a two-camera setup, where we merge the halves of the transformed camera images that do not contain their respective vanishing point, as illustrated in Figure 4.8. The use of two cameras at the same time results in improved image quality, as our view of the needed reflection surfaces and the associated light intensities is now available at a better angle.

Merging the two different input views, however, comes at a cost. The transformed images must be carefully aligned to create a uniform output image, but inaccuracies in the camera positions may result in image distortions. Therefore external camera-sphere-projector calibration is more of a recommendation rather than a luxury. For our prototypes we settled for a soft blending approach to mask such artifacts.


(a) A bottom view as seen by a typical user in a meeting around a white desk.

(b) A top view, the spherical mirror reflecting a hallway with staircase.

(c) Remapped image to be projected on the spherical screen in (a).

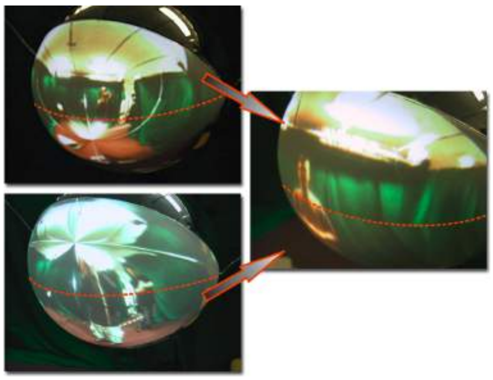
(d) Projection of the remapped image of the hallway captured in (b).

**Figure 4.7:** Our many-to-many prototype seen from various viewpoints in various scenarios.

## 4.7 Requirements Evaluation

We evaluate to what degree the presented solution adheres to the requirements laid out in the problem statement in section 1.1, p. 3. The scores correspond to the scale defined there and are plotted in Figure 4.10.

**Eye Contact** Because unfolding the environmental reflection captured on a (relatively small) specular sphere yields omnidirectional imagery with a projection center located at the center of that sphere, the user looks directly into the camera and eye contact is inherently guaranteed. However, image quality and pixel resolution are obvious limiting


(a) Two views blended into a single image.

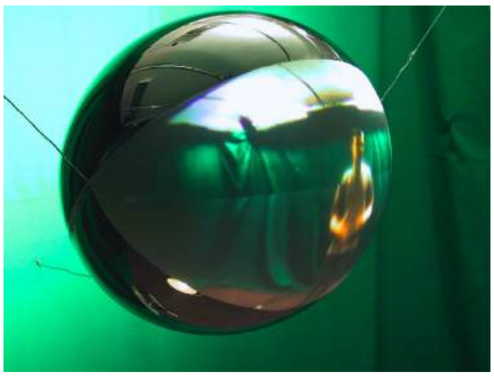
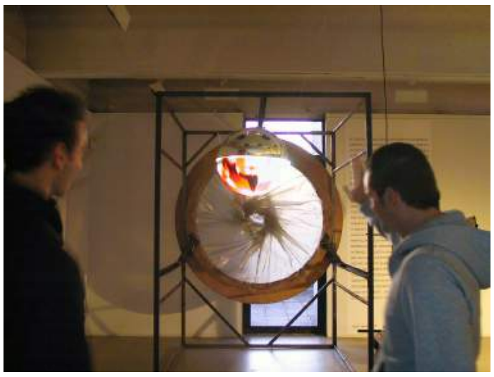

(b) Our one-to-one prototype in action.

**Figure 4.8:** For the one-to-one prototype, the halves of the transformed camera images that do not contain their respective vanishing point are merged and soft blending is employed to mask artifacts.


(a) A JanusLight located at
the top floor of a local theater.

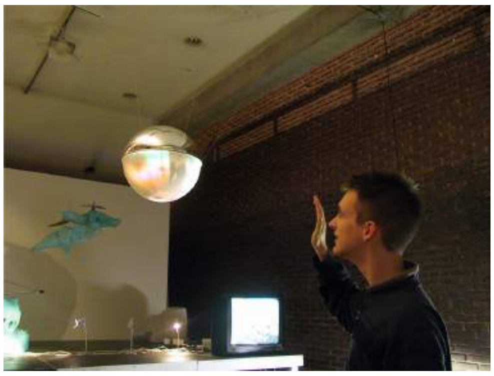
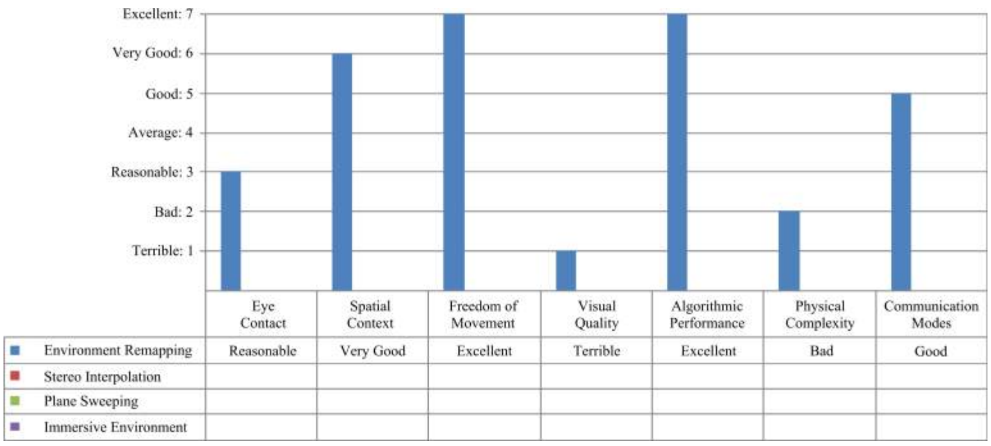

(b) A linked JanusLight located at
the bottom floor of the same theater.

**Figure 4.9:** Many-to-many JanusLights can be used as decorative lighting, offering a point of view into (network-) linked spaces, e.g. pubs, halls, theaters, exhibitions, events, etc.

factors. Nevertheless, when pixel resolution was high enough to see the eyes of the remote user, we have witnessed eye gaze to be accurate. **3/7 (reasonable)**

**Spatial Context** The spherical nature of the display screen unveils the full spatial context of the users. Compared with the limited information provided by a 2D viewing window, we believe this to be a significant enhancement. However, the displaying sphere remains relatively small. **6/7 (very good)**



| | Eye Contact | Spatial Context | Freedom of Movement | Visual Quality | Algorithmic Performance | Physical Complexity | Communication Modes |
|---|---|---|---|---|---|---|---|
| 🟦 Environment Remapping | Reasonable | Very Good | Excellent | Terrible | Excellent | Bad | Good |
| 🟥 Stereo Interpolation | | | | | | | |
| 🟩 Plane Sweeping | | | | | | | |
| 🟪 Immersive Environment | | | | | | | |

**Figure 4.10:** Requirements evaluation of our environment remapping prototype for eye gaze correction. The scores correspond to the scale defined in section 1.1, p. 3. Missing data will be filled in as more solutions are developed in the next chapters.

**Freedom of Movement** In the many-to-many configuration, the local user receives an unprecedented freedom to move around in the entire 360-degree space surrounding his own JanusLight, while never loosing sight of the remote user and maintaining eye contact at any moment. The one-to-one configuration is slightly more limited, yet still offers considerable freedom of movement in front of the JanusLight. **7/7 (excellent)**

**Visual Quality** It is clear that the very low image quality proves to be the Achilles' heel of our JanusLights system, severely limiting eye contact and spatial context. Although we are currently unable to verify this in practice, we postulate that this is mainly due to limitations of the mathematical model and the prototypes' off-the-shelf components rather than the concept itself. **1/7 (terrible)**

**Algorithmic Performance** The view synthesis is completely independent of the scene structure. It does not require the recovery of the depth of the scene, but instead relies on the mathematical equations that map the captured input to the projected output, both interpreted as parallel rays of light under an affine camera model. This pixel-to-pixel mapping can be precomputed, which allows for an extremely lightweight implementation that easily operates at a real-time rendering speed on any contemporary GPU and even CPU. **7/7 (excellent)**

**Physical Complexity** Perhaps not the most convenient physical setup, since larger spheres requires a larger camera/projector to sphere distance. This may especially pose a problem if a large viewing surface is desired. A small viewing surface further limits eye contact, spatial context and freedom of movement. **2/7 (bad)**

**Communication Modes** Although the solution is best suited for many-to-many communication, one-to-one and even limited multi-party communication (by subdividing the many-to-many projection) are supported. **5/7 (good)**

## 4.8 Conclusion

We presented a novel camera-projection system for one-to-one and many-to-many video conferencing based on the capture of omnidirectional video from a spherical mirror combined with display on a spherical surface.

Eye gaze correction is achieved using only one capturing camera (at least in the many-to-many user scenario) and without any need to perform computationally expensive view interpolation. Instead, synthesizing the eye gaze corrected image is completely independent of the scene structure. It does not require the recovery of the depth of the scene, but instead relies on the mathematical equations that map the captured input to the projected output, both interpreted as parallel rays of light. This pixel-to-pixel image transformation has to be precomputed only once, resulting in a very fast and lightweight implementation.

Because unfolding the environmental reflection captured on a (small, relative to the environment) specular sphere yields omnidirectional imagery with a projection center located at the center of the sphere, the user looks directly into the camera when looking at the sphere and eye contact is inherently guaranteed. Moreover, the omnidirectional nature of the device allows many users to communicate simultaneously, while the spherical screen unveils their full spatial context. They also possess an unprecedented freedom of movement in the entire space around the device.

Image quality suffers because of limitations of the mathematical model and the use of off-the-shelf hardware components (e.g. imperfect store surveillance mirrors). In spite of these limitations, we were still able to attain promising results and show it to be a functional concept.

### 4.8.1 Future Work

The image remapping equations are governed by an affine camera model. Using perspective cameras might complicate calculations to some extent, but it should be able to alleviate the vanishing point artifacts to some degree. Any missing image information at the vanishing point coordinates can then be interpolated.

We also assume a known external calibration of the camera-sphere-projector configuration. Achieving this alignment proves to be an elaborate manual process. An automatic calibration would decrease the setup time and at the same time would remove image distortions originating from misalignment. The work of Svoboda [2000], Pajdla et al. [2001] and Francken et al. [2007] is a good place to start.

Chapter 5

# Stereo Matching

## Contents

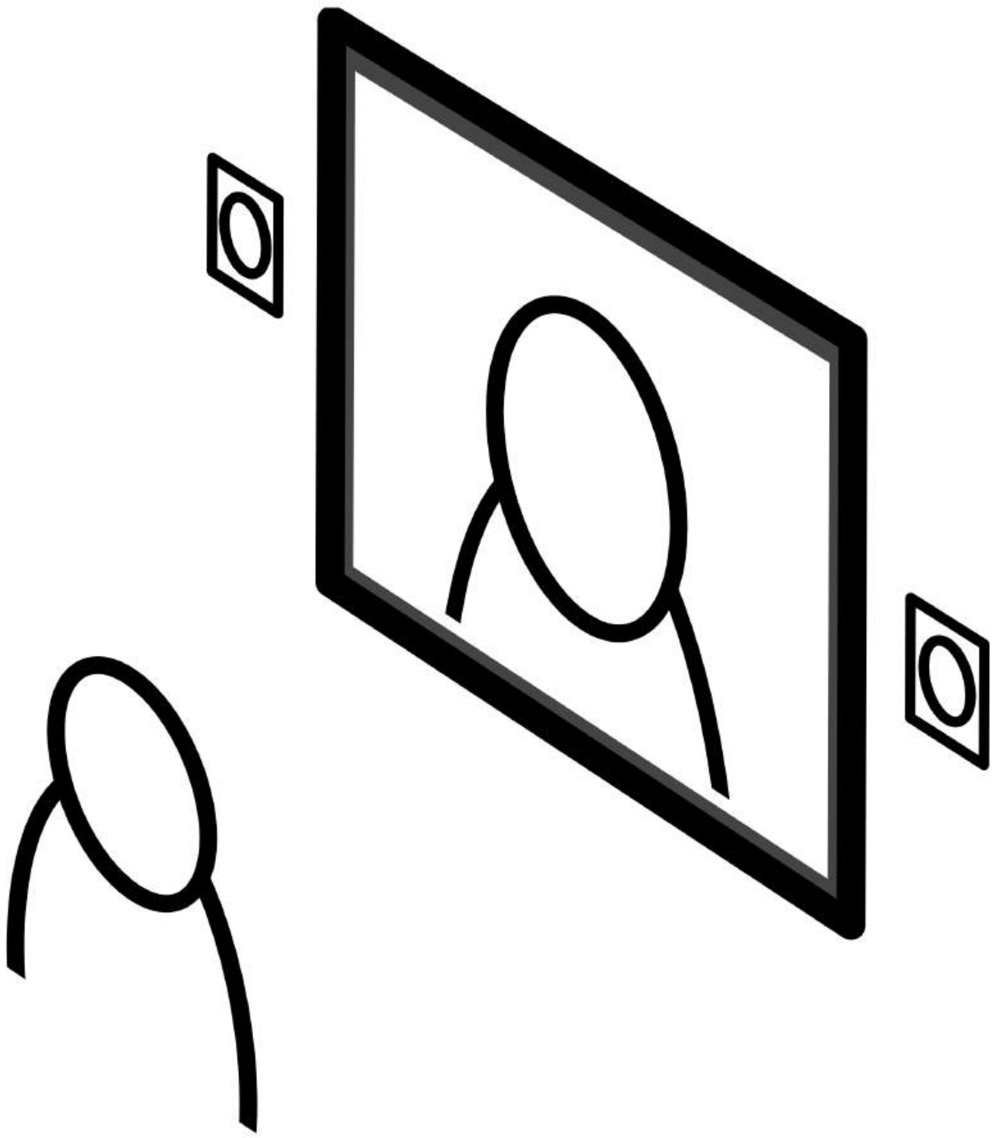


**Figure 5.1:** This chapter develops a novel stereo matching algorithm. In chapter 6, this algorithm will form an integral part of the two-camera solution depicted in the overview in Figure 1.1(b), p. 2.

Stereo matching takes a pair of images, estimates the apparent movement of each pixel from one image to the next and expresses this movement in a disparity map for the image under consideration. In chapter 6 we will correct eye gaze by interpolating between two cameras in a rectified stereo configuration. As stereo matching is the basis for stereo interpolation, one of the more common approaches to view interpolation in the literature, chapter 6 will rely heavily on the stereo matching algorithm that we develop here. Both chapters together constitute the two-camera solution in the overview in Figure 1.1(b), p. 2, repeated in Figure 5.1. We conceptually explained both and placed them in the context of depth-image-based rendering with implicitly determined geometry in section 2.1.2, p. 14.

We start this chapter with an in-depth overview of stereo matching research in section 5.1. As we will find out there, local disparity estimation algorithms typically consist of four stages: cost calculation, cost aggregation, disparity selection and disparity refinement [Scharstein and Szeliski, 2002]. Our main contribution in this chapter lies in the refinement stage [Dumont et al., 2014c, 2015]. In section 5.4 we present an iterative disparity refinement method to significantly improve the quality of any initially estimated disparity map. One iteration of the refinement consists itself of four strictly defined stages: a disparity cross-check (section 5.4.1), bitwise fast voting (section 5.4.2), invalid disparity handling (section 5.4.3) and median filtering (section 5.4.4). We will also observe that the iterative process tends to converge to a final solution.

Our refinement depends heavily on local support windows that we first define in section 5.2. As a second contribution, we construct two edge-sensitive windows around the currently considered pixel that adapt their shape to the underlying color information in the input images. Hereby we assume that pixels with similar colors belong to the same object and therefore should get the same disparity value (or depth in the scene). Each window favors a specific edge direction. One window grows in the horizontal direction and stops at edges, and likewise the other window grows in the vertical direction. Unlike the method of Zhang et al. [2009a] which uses only a horizontal window, we combine these two directions so that vertical edges are not favored.

Although applicable to a disparity map that was computed using any disparity estimation algorithm, the ultimate success of our refinement still depends on the quality of the initial disparity map. In our third contribution we therefore develop a novel disparity map estimation algorithm. It is presented in section 5.3 and covers the other three stages previously mentioned: cost calculation (section 5.3.1), cost aggregation (section 5.3.2) and disparity selection (section 5.3.3). This time the novelty lies in the cost aggregation stage, where we combine the edge-sensitive local support windows from section 5.2 into a global support window for increased disparity hypothesis confidence [Dumont et al., 2014b, 2015].

We demonstrate the effectiveness of our stereo matching method on various standard Middlebury datasets [Scharstein and Szeliski, 2003] in section 5.5. First we quantitatively and qualitatively compare our iteratively refined disparity maps with their ground truth (section 5.5.1), after which we take a look at processing time (section 5.5.2). As local pixel-wise algorithms map very well to parallel hardware, we achieve real-time performance by implementing the entire stereo matching pipeline in CUDA [Sanders and Kandrot, 2010; NVIDIA Corporation, 2007]. This exposes the GPU as a massive pool of directly programmable parallel threads [Ryoo et al., 2008; Goorts et al., 2009, 2010], as explained in section 2.3.2, p. 23, and depicted in Figure 2.8, p. 24. We finally conclude in section 5.6.

Throughout this chapter, we illustrate our stereo matching algorithm on the Middlebury Teddy scene. Both its left and right color images, together with their ground truth disparity maps, are shown in Figure 5.2. The full algorithmic chain is quite extensive, but an overview is given in Figure 5.3. The left ($I$) and right ($I'$) color images serve as input to this chain (step (a)) and our goal now becomes to estimate disparity maps that are as close as possible to their left ($D_{GT}$) and right ($D'_{GT}$) ground truth.

## 5.1 Related Work

Stereo matching algorithms for dense disparity estimation can typically be classified as either local or global. The stereo matching algorithm developed in this chapter is characterized as local, as it restricts itself to matching local neighborhoods around pixels. Scharstein and Szeliski [2002] break down stereo matching in a clear and concise taxonomy, with four well-defined stages: cost calculation, cost aggregation, disparity selection and disparity refinement. Later, Rogmans et al. [2009c] formalize the taxonomy introduced by Scharstein [1996] for intermediate view synthesis.

Local stereo matching algorithms rely heavily on good and efficient cost aggregation, where the matching costs of neighboring pixels are taken into account to acquire a more confident cost. The early conventional approach was to use fixed-size square windows, where at best the support weight for each pixel in the window tapers off gradually, according to its geometric proximity to the center pixel [Nishihara, 1984; Faugeras et al., 1993]. While extremely fast and straightforward to implement, the result is often noisy and contains severe artifacts. This is because the naive local approach does not integrate any scene knowledge at all, as it implicitly assumes that all pixels in the square window have similar disparity value as the center pixel does. Furthermore, the ideal window size may vary within a single image: larger ones for weakly textured areas and smaller ones for areas with finer detail. Consequently, local stereo matching very easily suffers from edge fattening, a phenomenon that manifests itself as *leakage* of disparities over object (occlusion) edges and depth discontinuity boundaries. The general consensus in attempting to counteract these nefarious properties, is to adapt the aggregation window to the underlying information in the reference images by varying its size, shape, location, support weights, or a mixture of all those.

A common approach is to choose from a range of predefined rectangular window sizes [Kanade and Okutomi, 1994; Boykov et al., 1998]. Yang and Pollefeys [2003] implement a multi-resolution hierarchical approach to combine cost measurements for square windows of varying sizes on commodity graphics hardware. Additionally, Veksler [2003] efficiently

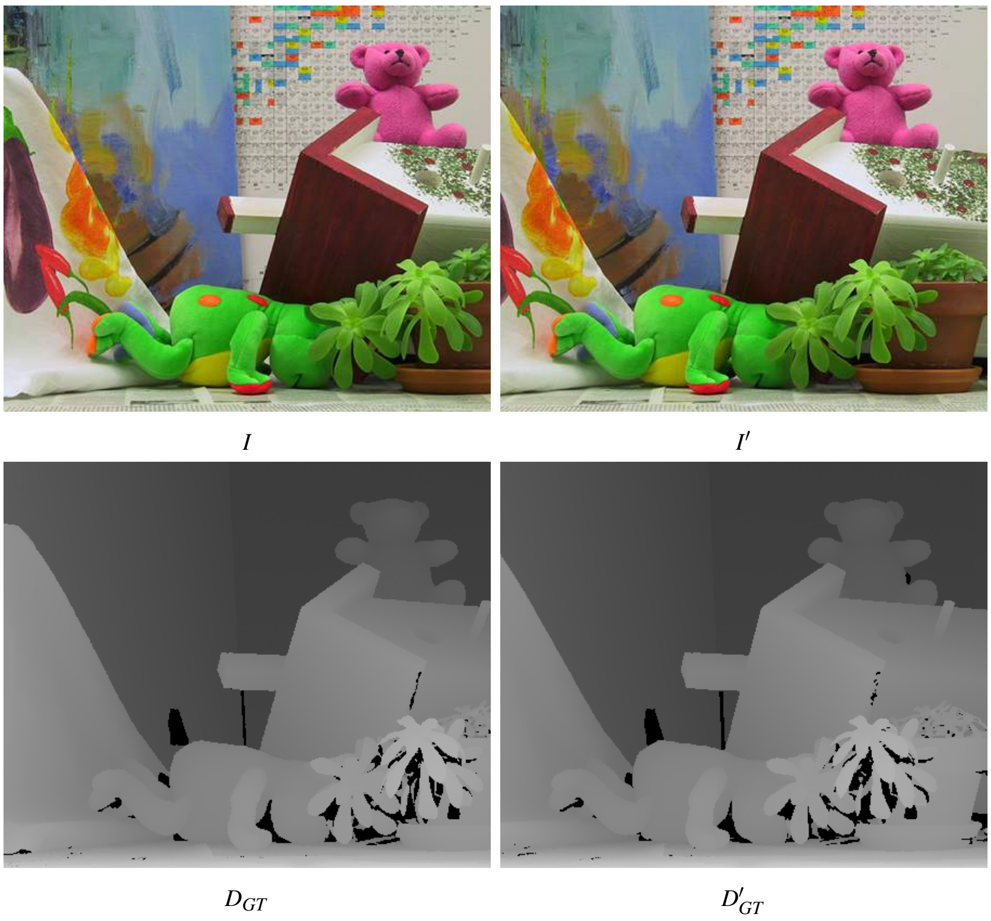


**Figure 5.2:** The Teddy scene of the standard Middlebury dataset [Scharstein and Szeliski, 2003], which will serve as a running example throughout this chapter. Our goal is to estimate as well as possible the left ($D_{GT}$) and right ($D'_{GT}$) ground truth disparity maps from the left ($I$) and right ($I'$) input images. Black patches in the ground truth disparity maps indicate missing data.

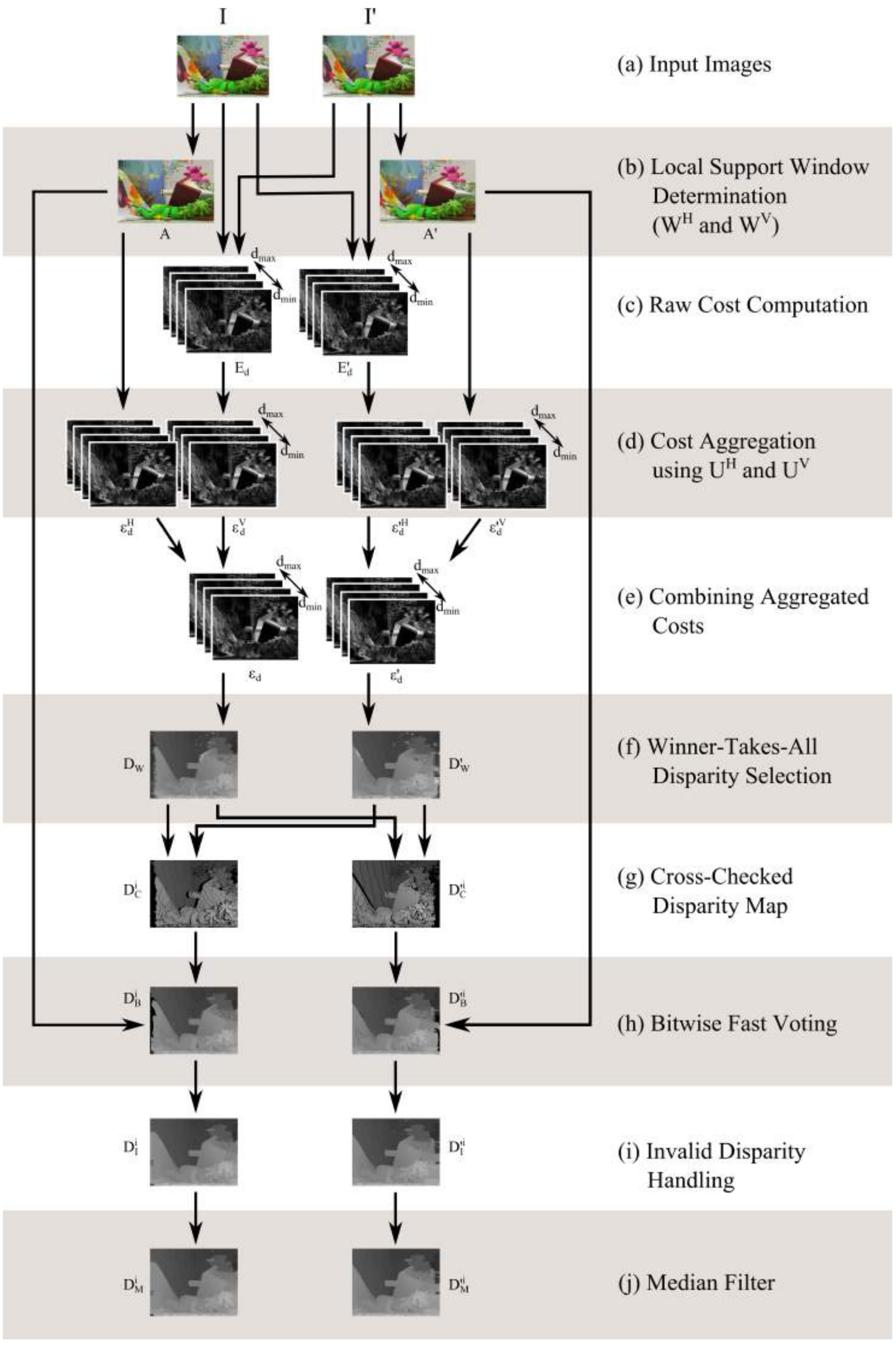
aggregates costs over those windows by using the integral images technique, a technique that we will also adopt in section 5.3.2.1.

Using shiftable windows entails placing multiple windows at different locations (not just centered around the pixel that we are trying to match) and selecting the one that produces the smallest matching cost [Fusiello and Roberto, 1997; Bobick and Intille, 1999; Kang et al., 2001; Hirschmüller et al., 2002]. Yang et al. [2004a] exploit the GPU's bilinear sampling functionality to efficiently aggregate matching costs over six different window shapes surrounding the pixel of interest. Another approach is to take an isotropic 2D kernel, truncate it into its four constituent parts (upper/lower, left/right) and use hierarchical combinations of those parts to aggregate costs [Lu et al., 2007, 2009]. It combines the advantages of large support windows and shiftable windows, while also being moderately edge-aware and providing support weights on geometric proximity.

Rectangular windows struggle with arbitrarily shaped depth discontinuities. However, in many cases we may assume that image patches with similar color belong to the same object and therefore have the same depth in the scene. In other words, the color discontinuity boundaries in the images are often also the depth discontinuity boundaries in the scene. Based on this property, segmentation-based approaches select the sizes and shapes for cost aggregation windows accordingly. Both methods that incorporate color segmentation [Tao and Sawhney, 2000; Bleyer and Gelautz, 2005; Gerrits and Bekaert, 2006; Tombari et al., 2008; Wang and Zheng, 2008] and edge detection [Gong and Yang, 2005] have received attention. These methods mostly struggle with representing irregularly shaped aggregation windows in an efficient manner.

Instead of changing the size or shape of the aggregation windows, a recent development with promising results is to adapt the support weights in fixed-size windows. Commonly known as bilateral filters, they were originally developed as edge-preserving and noise-reducing smoothing filters for images [Tomasi and Manduchi, 1998; Durand and Dorsey, 2002; Chen et al., 2007]. In essence, the support weights depend not only on the geometric distance between pixels, but also on the photometric difference [Ansar et al., 2004; Yoon and Kweon, 2006]. An approximated bilateral grid, extended with aggregation in the temporal domain, is presented by Richardt et al. [2010], whereas Tombari et al. [2007] incorporate segmentation information. Hosni et al. [2009] improve on the photometric distance of the bilateral filter by using the geodesic distance. For a neighboring pixel to obtain a high support weight, there must be a path to the center pixel along which the color does not change significantly. Recently, the bilateral filter has been redefined as a special case of the guided filter by

**Figure 5.3** ***(facing page)*****:** Overview of our stereo matching method. (a) The input is a pair of rectified stereo images, in this case the standard Middlebury dataset Teddy from Figure 5.2. (b) First, a horizontal and vertical edge-sensitive local support window is constructed in each image, based on color consistency. (c) Next, a pixel-wise cost per disparity is computed. (d) These costs are aggregated over the support windows. There are two windows and thus two sets of aggregated costs. (e) The costs of the two windows are combined. (f) The most suitable disparity value is selected by a winner-takes-all (WTA) approach. (g) A cross-check is performed to find any mismatches, typically caused by occlusions around edges. (h) The local support windows are used again to let the bitwise fast voting determine the most occurring disparity value in each window. (i) Finally, any remaining invalid disparities are filled in by looking for the nearest valid disparity on the same scanline, (j) and a median filter is applied to remove noise. The steps (c)–(f) estimate the initial disparity map and are executed only once. The steps (g)–(j) constitute the iterative disparity refinement and can be repeated multiple times. The result is two dense disparity maps, one for each input image.


I
I'
(a) Input Images
A
A'
(b) Local Support Window Determination ($W^H$ and $W^V$)
$d_{max}$
$d_{min}$
(c) Raw Cost Computation
$E_d$
$E'_d$
(d) Cost Aggregation using $U^H$ and $U^V$
$\varepsilon_d^H$
$\varepsilon_d^V$
$\varepsilon_d'^H$
$\varepsilon_d'^V$
(e) Combining Aggregated Costs
$\varepsilon_d$
$\varepsilon'_d$
$D_W$
$D'_W$
(f) Winner-Takes-All Disparity Selection
$D_C^i$
$D_C'^i$
(g) Cross-Checked Disparity Map
$D_B^i$
$D_B'^i$
(h) Bitwise Fast Voting
$D_I^i$
$D_I'^i$
(i) Invalid Disparity Handling
$D_M^i$
$D_M'^i$
(j) Median Filter

He et al. [2010]. Rhemann et al. [2011] let this filter smooth matching costs in 2D $(x,d)$ slices of the 3D $(x,y,d)$ disparity space image (a.k.a. matching cost volume). Since this filter is well behaved near edges, the edge fattening effect is largely avoided. Moreover, the guided filter can transfer structure of the reference image to the filtered output, enabling new applications beyond stereo matching. Unfortunately, the non-linearity and non-separability of bilateral filters – every support weight is uniquely dependent on its center pixel – results in high memory consumption and computational complexity. It remains challenging to obtain accurate real-time performance, even on contemporary graphics hardware [Yang, 2014]. Hosni et al. [2013] evaluate the performance of various methods for computing adaptive support weights.

In our work, we construct two edge-sensitive support windows around the current pixel of interest. Instead of adapting the windows' weights, we adapt their shape to the underlying color discontinuity boundaries in the reference images. Each window favors a specific edge direction. One window grows in the horizontal direction and stops at edges, likewise the other window grows in the vertical direction. Opposite to the method of Zhang et al. [2009a], which uses only a horizontal window, we combine two directions, so that vertical edges are not favored. Next, our windows are employed not only in the cost aggregation (section 5.3.2), but also in the refinement stage (section 5.4). This last stage is often regarded as an optional post-processing stage. Even more so in real-time environments, where low-complexity refinement or no refinement at all is generally favored. We, however, argue that the refinement stage is of equal – if not more – importance and is best not to be ignored. We formulate a four-step iterative refinement process that is able to reliably correct mismatches and fill in occlusions, while taking into account the challenging depth discontinuities.

Overall, the proposed method has some unique advantages. It is much easier to implement and more memory efficient than many leading real-time methods. The windows' shape is determined at run-time (without complex a priori color segmentation) and efficiently represented by a single quadruplet (the extent of two perpendicular axes, see section 5.2). We thereby not only avoid a complex window representation, but also additional computational complexity that comes from support weight dependency on the center pixel. Moreover, the compact representation is ideally suited for massive parallel processing on the GPU.

Alternatively to local stereo matching methods, global methods generally aim to optimize a global energy function of one form or another. They do not necessarily follow the aforementioned four stages and are based on (or include a combination of) graph cuts [Kang et al., 2001; Kolmogorov and Zabih, 2001; Deng et al., 2007; Papadakis and Caselles, 2010], belief propagation [Sun et al., 2003, 2005; Yang et al., 2006a, 2009, 2010a], dynamic programming [Forstmann et al., 2004; Kim et al., 2005; Wang et al., 2006b], segmented patches [Bleyer and Gelautz, 2005; Klaus et al., 2006; Zitnick and Kang, 2007; Yang et al., 2008], spatiotemporal consistency [Davis et al., 2005; Shechtman et al., 2005; Gong, 2006], structured light [Scharstein and Szeliski, 2003; Hermans, 2011] and numerical minimization methods [Min and Sohn, 2008].

We specifically develop our stereo matching algorithm to be very suitable for implementation on the GPU and thereby achieve high online processing speed. Another popular – easy to embed but less flexible – approach to achieve real-time performance is to implement on FPGAs (field programmable gate arrays) [Darabiha et al., 2006; Lazaros et al., 2008; Ambrosch et al., 2009; Banz et al., 2010; Zhang et al., 2011b].

Finally, all stereo matching algorithms discussed here determine dense correspondences in small-baseline setups. Wide-baseline matchers do exist, but they generate only sparse point-to-point correspondences [Pritchett and Zisserman, 1998; Tuytelaars and Van Gool, 2000; Lowe, 2004; Matas et al., 2004; Strecha et al., 2004; Mikolajczyk and Schmid, 2005; Mikolajczyk et al., 2005; Bay et al., 2008].

For an in-depth overview and comparison of real-time local stereo matching algorithms, refer to Scharstein and Szeliski [2002], Wang et al. [2006a], Gong et al. [2007], Rogmans et al. [2009c], Szeliski [2010] and Min et al. [2011].

## 5.2 Edge-Sensitive Local Support Windows

We first explain how to construct two edge-sensitive local support windows. They will be used both during the initial disparity map estimation in section 5.3 and during the iterative refinement in section 5.4.

For every pixel $p$ of the left image $I$, we first determine a horizontal axis $\mathbb{H}(p)$ and vertical axis $\mathbb{V}(p)$ crossing in $p$. These two axes can be represented as a quadruplet $\mathbb{A}(p)$:

$$\mathbb{A}(p) = (h_p^-, h_p^+, v_p^-, v_p^+) \tag{5.1}$$

where the component $h_p^-$ represents how many pixels the horizontal axis extends to the left of $p$, $v_p^+$ represents how many pixels the vertical axis extends above $p$, and so forth. Some of these axis quadruplets are drawn as yellow crosses of their horizontal and vertical axes in Figure 5.4 and in the overview in Figure 5.3(b).

To determine each component of the axis quadruplet, we keep extending an axis until the difference between the center pixel $p$ and the outermost pixel $q$ becomes too large:

$$\max_{c \in \{r,g,b\}} |I_c(p) - I_c(q)| \leq \tau \tag{5.2}$$

where $I_c(p)$ is the red, green or blue color channel of pixel $p$ and $\tau$ is the threshold for color consistency. We also stop extending if the size exceeds a maximum predefined length $L$.

From these four components, two local support windows for pixel $p$ can be derived, referred to respectively as the horizontal local support window $W^H(p)$ and the vertical local support window $W^V(p)$. Both derivations are illustrated in Figure 5.5.

Let's start by constructing the horizontal window $W^H(p)$. First, we need to create its vertical axis based on the values of $v_p^-$ and $v_p^+$. We call this the primary vertical axis $\mathbb{V}(p)$. Next, we consider the values of $h_q^-$, and $h_q^+$ for each pixel $q$ on the primary vertical axis.

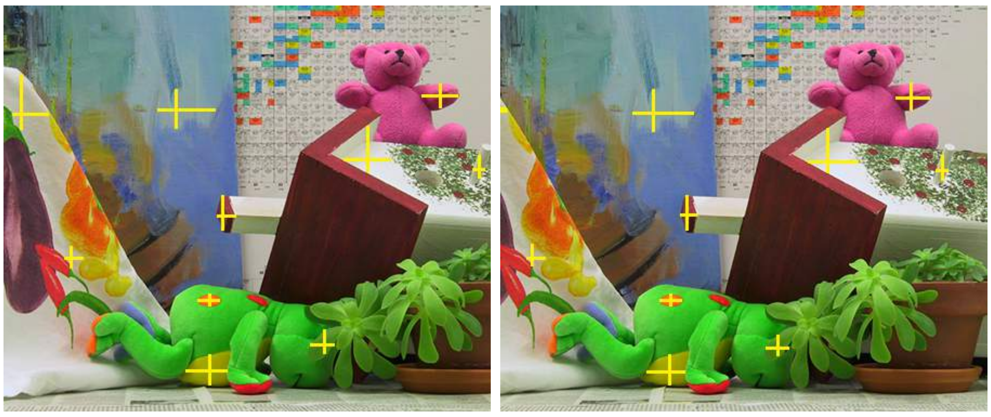


**Figure 5.4:** Some left ($\mathbb{A}$) and right ($\mathbb{A}'$) axis quadruplets from Equation 5.1, drawn as yellow crosses of their horizontal and vertical axes.

These define a horizontal axis per pixel $q$ on the primary vertical axis and are called the subordinate horizontal axes $\mathcal{H}(q)$. In short, this results in the orthogonal decomposition:

$$W^H(p) = \bigcup_{q \in \mathbb{V}(p)} \mathcal{H}(q) \tag{5.3}$$

Completely analogous, but in the other direction, we construct the vertical local support window $W^V(p)$ by creating a primary horizontal axis $\mathbb{H}(p)$ using $h_p^-$, and $h_p^+$, and on this axis creating the subordinate vertical axes $\mathcal{V}(q)$:

$$W^V(p) = \bigcup_{q \in \mathbb{H}(p)} \mathcal{V}(q) \tag{5.4}$$

To construct both windows for a center pixel $p$, we only require its single axis quadruplet $(h_p^-, h_p^+, v_p^-, v_p^+)$, together with the quadruplets that have been precomputed for every neighboring pixel. Thus memory usage and access is severely reduced, which is a serious consideration when using GPU computing.

Constructed this way, our windows are sensitive to edges in the image. The horizontal window $W^H(p)$ will fold nicely around vertical edges, because the width of each subordinate horizontal axis is variable. Horizontal edges are not followed as accurately, because the height of the window is fixed and only determined by its primary vertical axis. This situation, however, is reversed for the vertical window $W^V(p)$. Thus by using both windows, we do not favor a single edge direction, which will yield better results.

Finally, the notation $W'^H(p')$ and $W'^V(p')$ represents the local support windows for each pixel $p'$ in the right image $I'$.

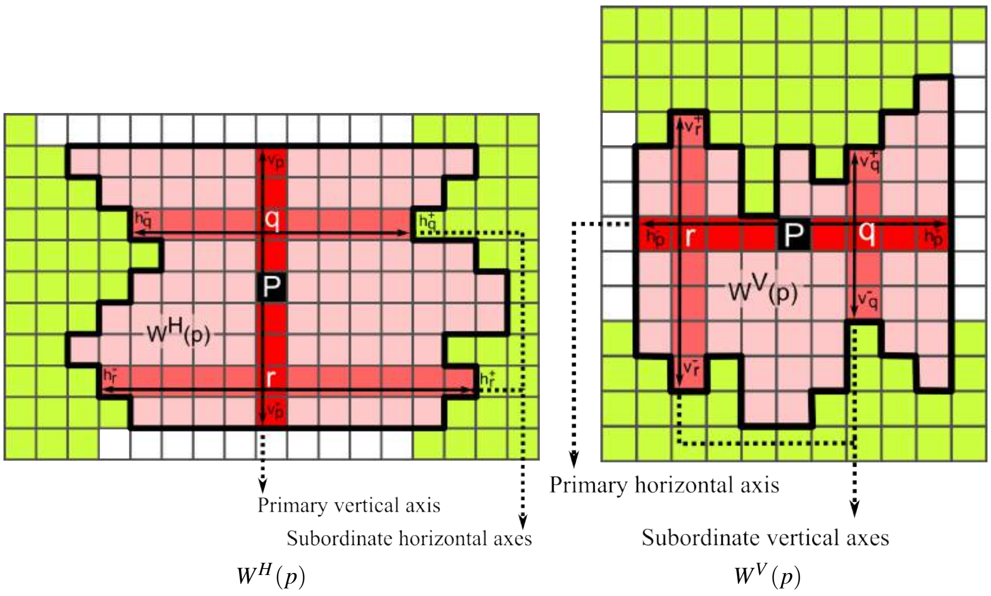


**Figure 5.5:** Derivation of the horizontal ($W^H(p)$, Equation 5.3) and vertical ($W^V(p)$, Equation 5.4) local support windows for pixel $p$, using its axis quadruplet $\mathbb{A}(p) = (h_p^-, h_p^+, v_p^-, v_p^+)$ of Equation 5.1.

## 5.3 Initial Disparity Estimation

In this section, our goal is to estimate an initial disparity map that will serve as input to our iterative refinement process in section 5.4. First, in section 5.3.1, we consider each disparity and calculate for each pixel in the left image the difference (i.e. matching cost) between that pixel and the corresponding pixel in the right image, based on the disparity under consideration. Next, in section 5.3.2, the costs of neighboring pixels are aggregated to obtain a more confident matching cost. Once the costs are aggregated per pixel and per disparity value, the most suitable disparity with the lowest cost is selected in section 5.3.3.

### 5.3.1 Per-Pixel Matching Cost

Let the range $R$ of valid disparity values $d$ be $R = [d_{min}, d_{max}]$. Then for a disparity hypothesis $d \in R$ and pixel $p$ of the left image $I$, consider the raw per-pixel matching cost $E_d(p)$, defined as the sum of absolute differences (SAD):

$$E_d(p) = \frac{\sum_{c \in \{r,g,b\}} |I_c(p) - I'_c(p')|}{e_{max}} \tag{5.5}$$

where pixel $p$ in the left image $I$ is compared with pixel $p'$ in the right image $I'$, and the coordinates of $p = (x_p, y_p)$ and $p' = (x_{p'}, y_{p'})$ relate to the disparity hypothesis $d$ as $x_{p'} =$

$x_p - d$, $y_{p'} = y_p$. The constant $e_{max}$ normalizes the cost $E_d(p)$ to the floating point range $[0,1]$. For example, when processing RGB images with 8 bits per channel, $e_{max} = 3 \times 255$.

We calculate $E_d(p)$ for each pixel $p$ and refer to $E_d$ as the per-pixel left confidence (or cost) map for disparity $d$. Similarly the per-pixel right confidence map $E'_d$ can be constructed by calculating $E'_d(p')$ for each pixel $p'$ analogously to Equation 5.5, with the x-coordinates of $p$ and $p'$ now related as $x_p = x_{p'} + d$. The left and right per-pixel confidence maps for disparity $d = d_{min}$ are shown in Figure 5.3(c).

## 5.3.2 Cost Aggregation over Global Support Windows

To reliably aggregate costs, we must simultaneously consider both local support windows $W(p)$ for pixel $p$ in the left image and $W'(p')$ for pixel $p'$ in the right image. If we only consider the local support window $W(p)$, the matching cost aggregation will be polluted by outliers in the right image and vice versa. Therefore, while processing for disparity hypothesis $d$, the two local support windows are combined into a global support window $U_d(p)$. Distinguishing again between horizontal and vertical support windows, they are defined as:

$$U_d^H(p) = W^H(p) \cap W'^H(p') \tag{5.6}$$

$$U_d^V(p) = W^V(p) \cap W'^V(p') \tag{5.7}$$

where the coordinates of $p = (x_p, y_p)$ and $p' = (x_{p'}, y_{p'})$ are again related to the disparity hypothesis $d$ as $x_{p'} = x_p - d$, $y_{p'} = y_p$. In practice, this simplifies beautifully to taking the component-wise minimum of their axis quadruplets from Equation 5.1:

$$\mathbb{A}_d(p) = \min\left(\mathbb{A}(p), \mathbb{A}'(p')\right) \tag{5.8}$$

Two more confident matching costs $\varepsilon_d^H(p)$ and $\varepsilon_d^V(p)$ can now be aggregated over each pixel $s$ of the horizontal and vertical global support windows $U_d^H(p)$ and $U_d^V(p)$ respectively:

$$\varepsilon_d^H(p) = \frac{1}{\|U_d^H(p)\|} \sum_{s \in U_d^H(p)} E_d(s) \tag{5.9}$$

$$\varepsilon_d^V(p) = \frac{1}{\|U_d^V(p)\|} \sum_{s \in U_d^V(p)} E_d(s) \tag{5.10}$$

where the number of pixels $\|U_d(p)\|$ in the support window acts as a normalizer. These aggregated confidence maps are shown in Figure 5.3(d) for disparity hypothesis $d = d_{min}$.

We next propose three methods to select the final aggregated cost $\varepsilon_d(p)$, based on $\varepsilon_d^H(p)$ and $\varepsilon_d^V(p)$. The first method tries to match as large as possible windows, assuming that larger windows are less error-prone:

$$\varepsilon_d(p) = \begin{cases} \varepsilon_d^H(p) & \text{if } \|W^H(p)\| \geq \|W^V(p)\| \\ \varepsilon_d^V(p) & \text{otherwise} \end{cases} \tag{5.11}$$

The second method uses a weighted sum and is more robust against errors in the matching process:

$$\varepsilon_d(p) = \gamma\, \varepsilon_d^H(p) + (1-\gamma)\, \varepsilon_d^V(p) \tag{5.12}$$

where $0 \leq \gamma \leq 1$ allows to steer toward scenes with primarily horizontal or vertical edges.

The third and final method takes the minimum and assumes that the absolute lowest cost is the correct solution:

$$\varepsilon_d(p) = \min\left(\varepsilon_d^H(p), \varepsilon_d^V(p)\right) \tag{5.13}$$

Again the combined confidence map $\varepsilon_d$ is shown in Figure 5.3(e).

The aggregation is repeated over the right image, which means computing $\mathbb{A}'_d(p') = \min(\mathbb{A}(p), \mathbb{A}'(p'))$, with $p$ and $p'$ now related as $x_p = x_{p'} + d$ and from there setting up an analogous reasoning to end up at the right aggregated confidence map $\varepsilon'_d$.

#### 5.3.2.1 Fast Cost Aggregation using Orthogonal Integral Images

From the global axis quadruplet $\mathbb{A}_d(p)$ of Equation 5.8 and following the same reasoning that defined the local support windows in section 5.2, an orthogonal decomposition of the global support windows $U_d^H(p)$ and $U_d^V(p)$ can be obtained analogously to Equation 5.3 and Equation 5.4:

$$U_d^H(p) = \bigcup_{q \in \mathbb{V}_d(p)} \mathcal{H}_d(q) \tag{5.14}$$

$$U_d^V(p) = \bigcup_{q \in \mathbb{H}_d(p)} \mathcal{V}_d(q) \tag{5.15}$$

This orthogonal decomposition is key to a fast and efficient implementation of the cost aggregation step [Crow, 1984; Veksler, 2003; Zhang et al., 2009a]. Substituting Equation 5.14 into Equation 5.9 and Equation 5.15 into Equation 5.10, we separate the inefficient $\sum_{s \in U_d(p)} E_d(s)$ into a horizontal and vertical integration:

$$\varepsilon_d^H(p) = \sum_{q \in \mathbb{V}_d(p)} \left( \sum_{s \in \mathcal{H}_d(q)} E_d(s) \right) \tag{5.16}$$

$$\varepsilon_d^V(p) = \sum_{q \in \mathbb{H}_d(p)} \left( \sum_{s \in \mathcal{V}_d(q)} E_d(s) \right) \tag{5.17}$$

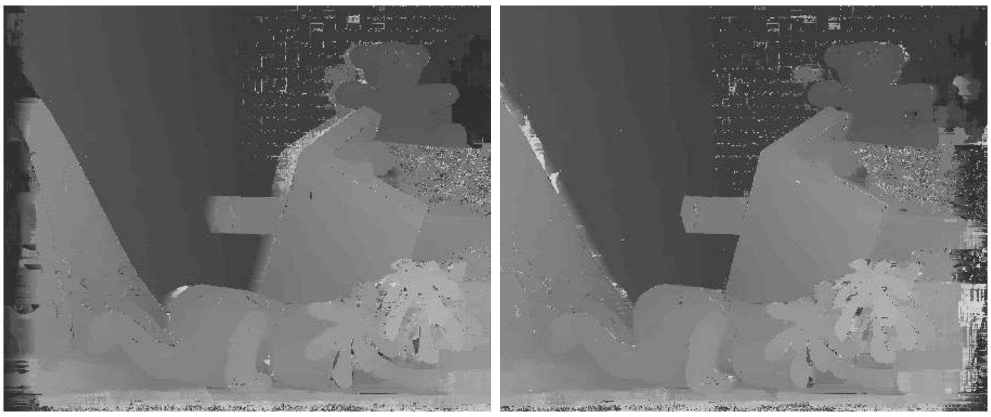


**Figure 5.6:** The left ($D_W$) and right ($D'_W$) winner-takes-all disparity maps, as determined by Equation 5.18.

where the normalizer $\frac{1}{\|U_d(p)\|}$ has been omitted for clarity.

For the global horizontal support window $U_d^H(p)$, Equation 5.16 intuitively means to first aggregate costs over its subordinate horizontal axes $\mathcal{H}_d(q)$ and then over its primary vertical axis $\mathbb{V}_d(p)$. Vice versa for the vertical configuration of $U_d^V(p)$ in Equation 5.17.

## 5.3.3 Disparity Selection

After the left and right aggregated confidence maps have been computed for every disparity $d \in R = [d_{min}, d_{max}]$, the best disparity per pixel (i.e. the one with lowest cost $\varepsilon_d(p)$) is selected using a winner-takes-all (WTA) approach:

$$D_W(p) = \arg\min_{d \in R} \varepsilon_d(p) \tag{5.18}$$

which results in the disparity maps $D_W$ for the left image and $D'_W$ for the right image, both shown in Figure 5.6 and in the overview in Figure 5.3(f). These disparity maps will serve as input to the iterative refinement process described next in section 5.4.

Finally, we also keep a horizontally and vertically aggregated confidence map:

$$\varepsilon^H(p) = \min_{d \in R} \varepsilon_d^H(p) \tag{5.19}$$

$$\varepsilon^V(p) = \min_{d \in R} \varepsilon_d^V(p) \tag{5.20}$$

## 5.4 Iterative Disparity Refinement

We now iteratively refine the two initial disparity maps $D_W$ and $D'_W$. One iteration consists of four stages, (g) to (j) in the overview in Figure 5.3.

First we cross-check the disparities between the two disparity maps in section 5.4.1. Next, the local support windows as described in section 5.2 are employed again to update a pixel's disparity with the disparity that appears most inside its windows. This method is the most powerful and is detailed in section 5.4.2. Any invalid disparities that remain after this are handled in section 5.4.3. In the last stage in section 5.4.4, the disparity map is median filtered to remove any remaining speckle noise. Finally, we initialize for the next iteration in section 5.4.5.

### 5.4.1 Disparity Cross-Check

A left-to-right cross-check means that for each of the pixels $p$ of the left disparity map $D_W$, the corresponding pixel $p'$ is determined in the right image based on the disparity value $D_W(p)$, and the disparity value $D'_W(p')$ in the right disparity map is compared with $D_W(p)$. If they differ, the cross-check fails and the disparity is marked as invalid. Introducing the superscript $i \geq 1$ to denote the current refinement iteration, this is expressed as:

$$D^i_C(p) = \begin{cases} D^{i-1}_W(p) & \text{if } D^{i-1}_W(p) = D'^{i-1}_W(p') \\ \mathit{INVALID} & \text{otherwise} \end{cases} \tag{5.21}$$

with $D^0_W = D_W$ and $D'^0_W = D'_W$, and with $p$ now related to $p'$ as $x_{p'} = x_p - D^{i-1}_W(p)$, $y_{p'} = y_p$. The process is then reversed for a right-to-left cross-check of the disparity map $D'^{i-1}_W$, which leaves us with the left and right cross-checked disparity maps $D^i_C$ and $D'^i_C$.

Invalid disparities are most likely to occur around edges in the image, where occlusions are present in the scene. These occluded regions are shown as pure black (marked as invalid) pixels in Figure 5.7 and in the overview in Figure 5.3(g).

### 5.4.2 Bitwise Fast Voting over Local Support Windows

This second stage updates a pixel's disparity with the disparity that is most present inside its local support windows $W^H(p)$ and $W^V(p)$ as defined in section 5.2. We may say that this is valid, because pixels in the same window have similar colors by definition and therefore with high probability belong to the same object and should have the same disparity. Confining the search to the local support windows also ensures that we greatly reduce the risk of edge fattening artifacts.

To efficiently determine the most frequent disparity value within a support window, we apply a technique called *bitwise fast voting* [Zhang et al., 2009b, 2011a] and adapt it to handle both horizontally and vertically oriented support windows. At the core of the bitwise

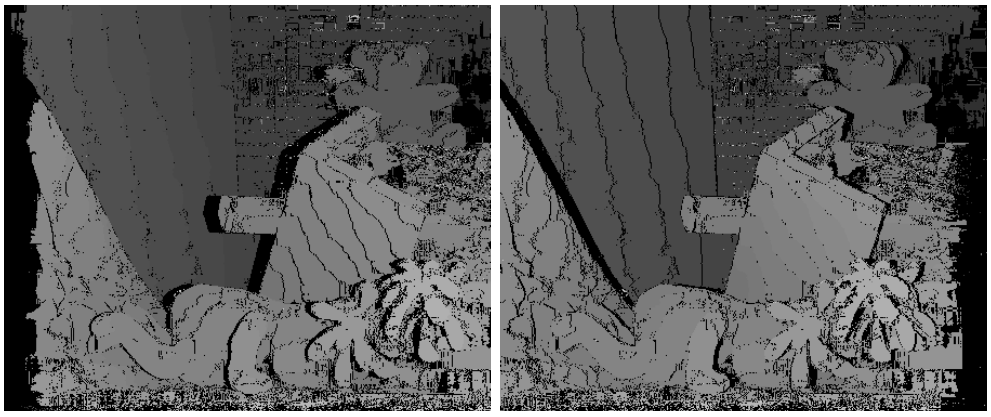


**Figure 5.7:** The left-to-right ($D_C^1$) and right-to-left ($D_C'^1$) cross-checked disparity maps, as determined by Equation 5.21. Black patches are disparities that have been invalidated by the cross-check.

fast voting technique lies a procedure that derives each bit of the most frequent disparity independently from its other bits.

First consider a pixel $p$ with local support window $W(p)$. We sum the $k^{\text{th}}$ bit $b_k(s)$ (either 0 or 1) of the disparity value $D_C^i(s)$ of all pixels $s$ in the support window and call the result $B_k(p)$ (for clarity, we drop the superscript $i$ for a moment). Furthermore distinguishing again between horizontal and vertical support windows, this gives:

$$B_k^H(p) = \sum_{s \in W^H(p)} b_k(s) \tag{5.22}$$

$$B_k^V(p) = \sum_{s \in W^V(p)} b_k(s) \tag{5.23}$$

The $k^{\text{th}}$ bit $D_B^k(p)$ of the final disparity value $D_B(p)$ is then decided as:

$$D_B^k(p) = \begin{cases} 1 & \text{if } B_k(p) > \beta \times N(p) \\ 0 & \text{otherwise} \end{cases} \tag{5.24}$$

where $\beta \in [0,1]$ is a sensitivity factor that we will come back to below. All this is illustrated in Figure 5.8.

We are left to determine exactly what $B_k(p)$ and $N(p)$ in Equation 5.24 are. For this we again propose three methods. The first method is similar to Equation 5.11 and assumes that the voting is more reliable over larger windows:

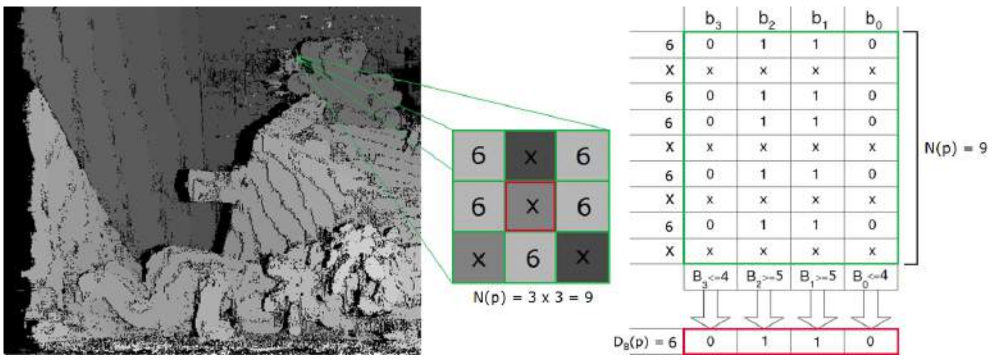


**Figure 5.8:** Bitwise fast voting aims to update a pixel's disparity with the most occurring value inside a local support window around that pixel. In this example, the disparities inside a square $3 \times 3$ window $W(p)$ are considered. The most occurring value happens to be 6 (binary 0110), thus we expect the center pixel $p$ to be updated to 6. X's may be any valid disparity, including 6. To calculate pixel $p$'s new disparity $D_B(p)$ efficiently, the bitwise fast voting sums every bit $b_k$ $(0 \leq k \leq 3)$ of every disparity (in green) into the counter $B_k$. In other words, $B_k$ counts how many bits on position $k$ are 1. If the amount of bits on position $k$ is overwhelmingly 1, the $k^{\text{th}}$ bit of pixel $p$'s new disparity $D_B(p)$ is also voted to be 1 (in red). Here, overwhelmingly means more than half of the window size $N(p) = 9$ (thus more than 4), but is really expressed by Equation 5.24 with $\beta = 0.5$. Also, instead of a simple square window, we use a combination of our horizontal and vertical windows $W^H(p)$ and $W^V(p)$.

$$B_k(p) = \begin{cases} B_k^H(p) & \text{if } \|W^H(p)\| \geq \|W^V(p)\| \\ B_k^V(p) & \text{otherwise} \end{cases} \tag{5.25}$$

$$N(p) = \max\left(\|W^H(p)\|, \|W^V(p)\|\right) \tag{5.26}$$

The second method uses a weighted sum:

$$B_k(p) = \gamma\, B_k^H(p) + (1-\gamma)\; B_k^V(p) \tag{5.27}$$

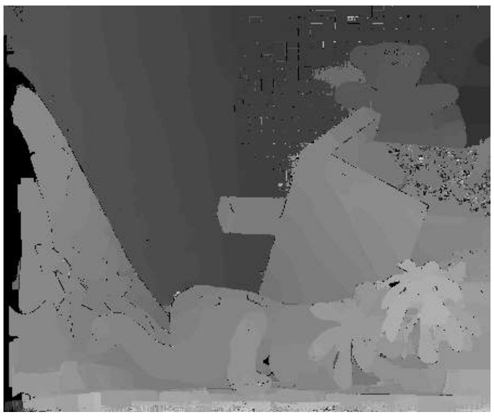
$$N(p) = \gamma\; \|W^H(p)\| + (1-\gamma)\; \|W^V(p)\| \tag{5.28}$$

where $\gamma$ is as in Equation 5.12.

The third and final method is similar to Equation 5.13:

$$B_k(p) = \begin{cases} B_k^H(p) & \text{if } \bar{\varepsilon}^H(p) \leq \bar{\varepsilon}^V(p) \\ B_k^V(p) & \text{otherwise} \end{cases} \tag{5.29}$$

$$N(p) = \begin{cases} \|W^H(p)\| & \text{if } \bar{\varepsilon}^H(p) \leq \bar{\varepsilon}^V(p) \\ \|W^V(p)\| & \text{otherwise} \end{cases} \tag{5.30}$$



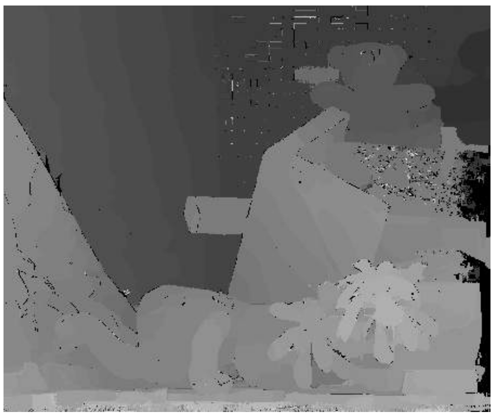


**Figure 5.9:** The left ($D_B^1$) and right ($D_B'^1$) disparity maps after bitwise fast voting, as determined by Equation 5.24 and following. Black patches are remaining invalid disparities that the bitwise vast voting was unable to fill in.

where $\bar{\varepsilon}(p)$ weights every bit vote $b_k(s)$ that counts toward $B_k(p)$ with its confidence value $\varepsilon(s)$ (as an extension to Equation 5.22 and Equation 5.23), and where we also need to differentiate again between the horizontal and vertical local support windows:

$$\bar{\varepsilon}^H(p) = \frac{\sum_{s \in W^H(p)} \left[b_k(s)\, \varepsilon^H(s)\right]}{\sum_{s \in W^H(p)} b_k(s)} \tag{5.31}$$

$$\bar{\varepsilon}^V(p) = \frac{\sum_{s \in W^V(p)} \left[b_k(s)\, \varepsilon^V(s)\right]}{\sum_{s \in W^V(p)} b_k(s)} \tag{5.32}$$

with either $b_k(s) = 0$ or $b_k(s) = 1$ (as defined earlier). This is the most precise yet most expensive method, because the computation of $\bar{\varepsilon}^H(p)$ and $\bar{\varepsilon}^V(p)$ requires an extra aggregation of the – already once aggregated – confidence maps $\varepsilon^H$ and $\varepsilon^V$ (Equation 5.19 and Equation 5.20) over the local support windows $W^H(p)$ and $W^V(p)$.

To recap, for a pixel $p$, Equation 5.24 states that the $k$th bit of its final disparity value is 1 if the $k$th bit appears as 1 in most of the disparity values under its local support window. The number of actual appearances of 1 are counted in $B_k(p)$, whereas the maximum possible appearances of 1 is given by the window size $N(p)$. Both $B_k(p)$ and $N(p)$ are determined by one of the three methods of Equations 5.25–5.30. The sensitivity factor $\beta$ controls how many appearances of 1 are required to confidently vote the result and is best set to 0.5.

It is important to note that certain disparities might be invalid due to the cross-check of $D_C^i(p)$ in section 5.4.1. While counting bit votes, we must take this into account by reducing $N(p)$ accordingly. This way the algorithm is able to update an invalid disparity by depending

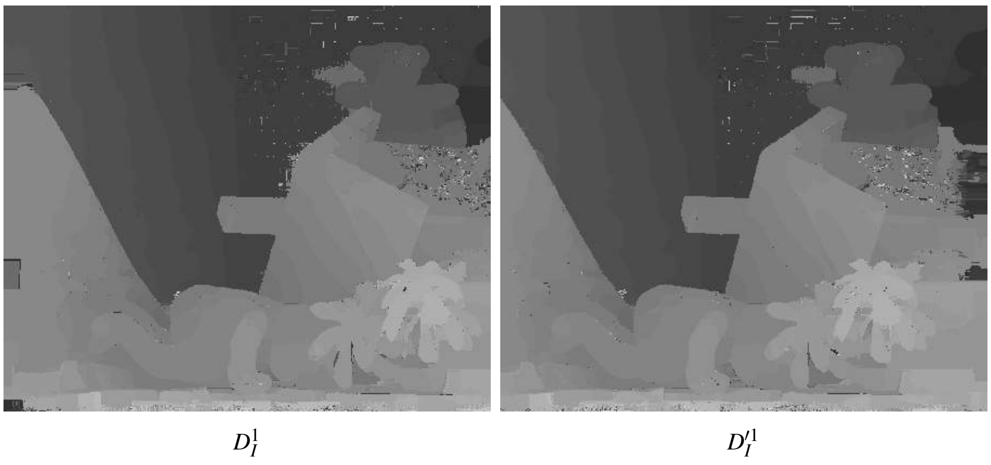


**Figure 5.10:** The left ($D_I^1$) and right ($D_I'^1$) disparity maps after any remaining invalid disparities have been filled in, as described in section 5.4.3.

on votes from valid neighbors only and thereby reliably fill in occlusions and handle part of the image borders.

Reintroducing the superscript $i$ for the $i^{\text{th}}$ iteration, the result of performing Equation 5.24 for all bits $k$ and all pixels $p$ is denoted as $D_B^i$. It is shown in Figure 5.9 and in the overview in Figure 5.3(h). The improvement in quality that this method yields in the disparity maps is also very apparent from the visual differences between Figure 5.6 to Figure 5.9.

A couple of key observations make that this method deserves to be called fast. First, the number of iterations needed to determine every bit of the final disparity value is limited by $d_{max}$. For example, in the Middlebury Teddy scene we use $d_{max}$ = 53, which is represented in binary as 110101, and thus only 6 iterations suffice. Moreover, in Figure 5.8 the disparity value 6 is represented in binary as 110 and thus calculating $B_3$ was in fact unnecessary. Furthermore, the votes can be counted very efficiently by orthogonally separating Equation 5.22 and Equation 5.23, analogously to Equation 5.16 and Equation 5.17. All this results in high efficiency with a low memory footprint.

### 5.4.3 Invalid Disparity Handling

The bitwise fast voting removes many invalid disparities by replacing them with the most occurring valid value inside their windows. It will fail, however, if the window does not contain any valid values, or in other words, if $N(p) = 0$ in Equation 5.24. This occurs mostly near the borders of the disparity maps, but can also manifest itself anywhere in the image where the occlusions are large enough.

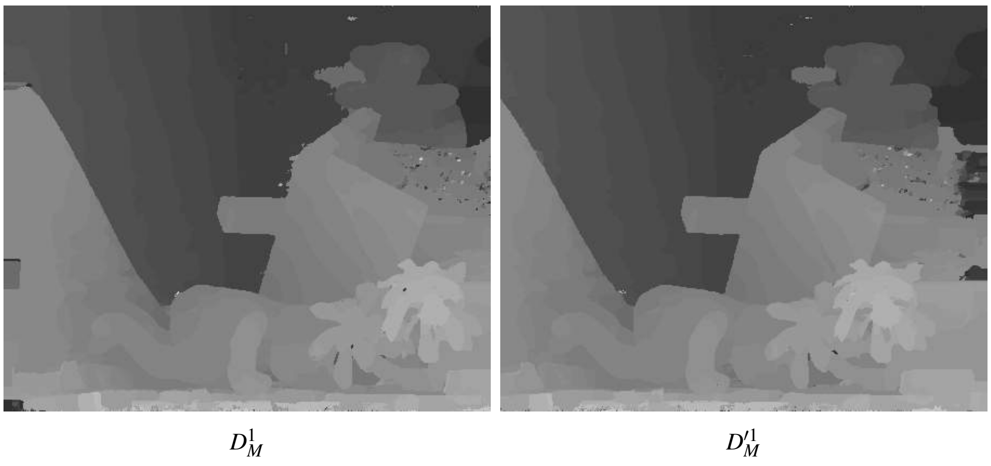


**Figure 5.11:** The left ($D^1_M$) and right ($D'^1_M$) disparity maps after application of a $3 \times 3$ median filter, as described in section 5.4.4. This completes one iteration of the disparity refinement process.

For each remaining pixel with an invalid disparity, we search to the left and to the right on its scanline for the closest valid disparity and store it in the corrected disparity map $D^i_I$. Unlike the bitwise fast voting, this scanline search is necessarily not confined to image patches of similar color. The result is shown in Figure 5.10 and in the overview in Figure 5.3(i).

### 5.4.4 Median Filter

In the last refinement step, small disparity outliers are filtered using a median filter. This results in the final disparity maps (for the current iteration) $D^i_M$ and $D'^i_M$, shown in Figure 5.11 and in the overview in Figure 5.3(j). A median filter has the property of removing speckle noise, in this case caused by disparity mismatches, while returning a sharp signal (unlike an averaging filter). We calculate the median for each pixel over a $3 \times 3$ window using a fast bubble sort implementation in CUDA [Astrachan, 2003].

### 5.4.5 The Next Iteration

This completes one iteration of the disparity refinement. The next iteration $i+1$ immediately starts again with the disparity cross-check of Equation 5.21 by setting $D^i_W = D^i_M$ and $D'^i_W = D'^i_M$ to obtain $D^{i+1}_C$ and $D'^{i+1}_C$. With each iteration the disparity map is considerably improved. In practice three to five iterations ($3 \leq i \leq 5$) suffice more often than not, at which point the refinement tends to converge to its final solution.

## 5.5 Results

We demonstrate the effectiveness of our method on the left viewpoint of various standard Middlebury datasets [Scharstein and Szeliski, 2003]. Section 5.5.1 performs a quantitative evaluation and discusses the effect of the iterative refinement, while section 5.5.2 takes a look at performance and processing time.

### 5.5.1 Quantitative Quality Evaluation

All quantitative measurements are expressed in dB PSNR (peak signal-to-noise ratio, where higher is better) compared with the respective scene's ground truth. Black patches in the ground truth disparity maps indicate invalid pixels (missing data) and are therefore not taken into account.

Our iterative refinement contributes significantly to the final quality of the disparity maps, as we will show in Figure 5.14 to Figure 5.17. Overall, many visual improvements are apparent, including the elimination of speckle noise, few errors at the image borders and sharply delineated edges with little to no edge fattening. All PSNR measurements are summarized in Table 5.1. From their plot in Figure 5.12, it is clear that the iterative refinement reaches its peak quality level after no more than three to five iterations, after which it tends to stabilize.

The effect of applying just one iteration of the refinement is already clear from the difference in visual quality for the Teddy scene in Figure 5.14. Without any refinement the result remains noisy, with a PSNR of 19.40 dB ($D_W^0$). Furthermore, the left border cannot be reliably matched and remains ambiguous, because this information is missing in the right image. One refinement iteration already increases the quality with 8 dB to 27.53 dB ($D_M^1$), yet some substantial noise overall and errors in the left border remain. A second iteration ($D_M^2$, 29.56 dB) resolves these issues for the most part and adds another 2 dB in PSNR. The next iterations take care of the last visually noticeable artifacts (e.g. the black erroneous patch in the lower left corner) and slightly better delineate the objects' edges, until the algorithm reaches its peak quality level at 29.96 dB for the fifth iteration ($D_M^5$). Performing any more iterations barely has any effect at all and the algorithm stabilizes on a final solution. One obvious erroneous patch remains next to the pink teddy's right ear. However, we postulate that this is due to the limited accuracy of the color consistency check that determines the local support windows (Equation 5.2), rather than a limitation of the refinement as a whole.

The Cones scene of Figure 5.15 is another challenging dataset that our iterative refinement is able to handle very well. Without refinement ($D_W^0$, 15.62 dB) the disparity map naturally remains noisy and one refinement iteration ($D_M^1$, 18.55 dB) is not able to improve the quality satisfactorily. A considerable amount of noise and errors remain, e.g. on the white cone in the background, on the little white box in the foreground, and in the left border. A second iteration is required to increase the quality with nearly 8 dB to 26.44 dB ($D_M^2$). The PSNR continues to slowly rise hereafter, until by the ninth iteration ($D_M^9$, 26.97 dB) even the mis-

| Fig. | Dataset | $D_W^0$ | $D_M^1$ | $D_M^2$ | $D_M^3$ | $D_M^4$ | $D_M^5$ | $D_M^6$ | $D_M^7$ | $D_M^8$ | $D_M^9$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5.14 | Teddy | **19.40** | **27.53** | **29.56** | 29.57 | 29.72 | **29.96** | 29.54 | 29.54 | 29.54 | 29.45 |
| 5.15 | Cones | **15.62** | **18.55** | **26.44** | 26.75 | 26.83 | 26.87 | 26.88 | 26.91 | 26.95 | **26.97** |
| 5.16 | Tsukuba | **20.04** | **24.16** | **24.18** | **24.30** | 24.15 | 24.09 | 24.06 | 24.05 | 24.05 | 24.04 |
| 5.17 | Venus | **19.76** | **25.04** | 28.97 | **30.83** | 30.88 | 31.06 | **31.10** | 31.08 | 31.10 | 31.09 |
| 5.18 | Sq. Win. | **18.21** | **23.53** | **26.44** | **28.10** | **28.03** | **27.94** | 27.83 | 27.74 | 27.69 | 27.65 |

**Table 5.1:** PSNR measurements in dB, for 1 ($D_M^1$) to 9 ($D_M^9$) iterations of our iterative refinement process from section 5.4 on the left disparity map of various Middlebury datasets [Scharstein and Szeliski, 2003]. The initial disparity map ($D_W^0$) has been computed with our stereo matching algorithm from section 5.3. The bottom entry (Sq. Win.) is an exception, where the initial disparity map of the Teddy dataset was computed using a conventional $17 \times 17$ square window and subsequently refined using our iterative refinement. Boldfaced numbers are referenced in the text and figures.

matches on the wooden framework in the background are fully resolved. During these many iterations, none of the other features in the disparity map are destroyed and all cones remain well discernible. Note in particular the outline of the green cone with blue base in front of the red cone. The cones in the left border disappear, because this information is again not available in the right image.

For the Tsukuba scene in Figure 5.16 the initial disparity map is rather noisy at 20.04 dB ($D_W^0$). In particular the orange lamp and the face statue in the foreground show signs of edge fattening (due to occlusions) at their left side. Just one refinement iteration ($D_M^1$, 24.16 dB) adds just over 4 dB and resolves most of these issues, but leaves some speckle noise in the background. After two more iterations the algorithm reaches its peak quality level at 24.30 dB ($D_M^3$), when even all mismatches between the arms of the orange lamp are resolved. Even so, the algorithm struggles to accurately match the background and tripod camera.

Venus in Figure 5.17 consists of three to four slanted planes with large homogeneously textured regions interspersed with rapidly changing fine – but similar – detail that may easily throw off most local window-based cost aggregation. However, after a few iterations the algorithm succeeds to comprehend the slanting of the planes and continues to refine it. It even surpasses the 30 dB frontier at the third iteration ($D_M^3$, 30.83 dB) and peaks three iterations later at 31.10 dB ($D_M^6$). However, the disparity map will always remain more coarse and lacks the smooth gradual change in gray-scale luminance of its ground truth ($D_{GT}$).

A great strength of our iterative refinement is that it can be applied to any local stereo matching algorithm, as long as the initial disparity map is of sufficient quality. To demonstrate this in the extreme case, we applied it to a disparity map that was computed using a conventional $17 \times 17$ square cost aggregation window. This causes a lot of edge fattening artifacts in the initial disparity map ($D_W^0$), as shown in Figure 5.18. Our refinement improves the disparity map dramatically with nearly 10 dB: from 18.21 dB for $D_W^0$ to 28.10 dB for $D_M^3$ after only three iterations. Nevertheless, all artifacts from a naive stereo matching algorithm cannot be eliminated, not even by applying many more iterations.

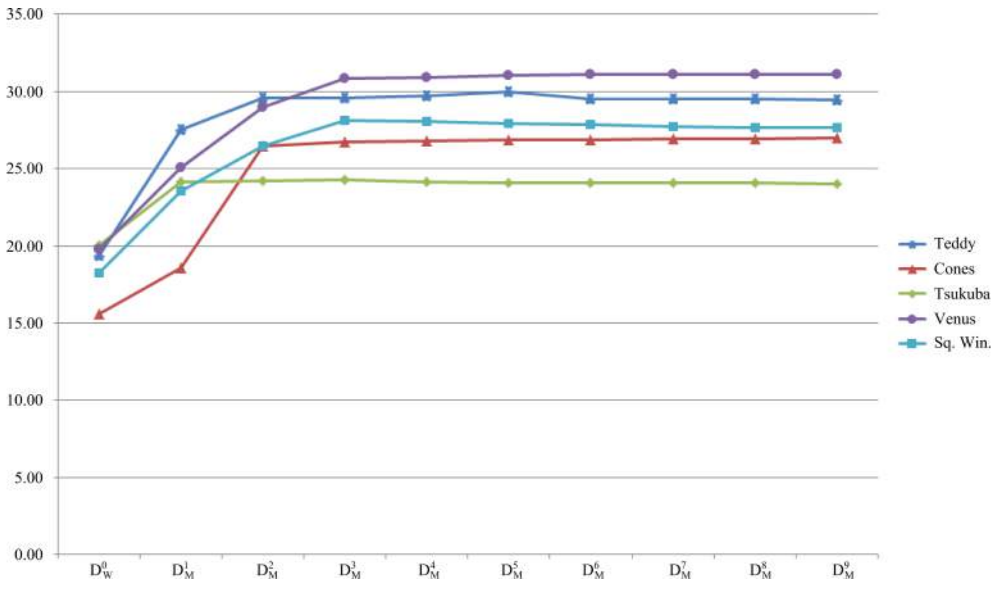


**Figure 5.12:** From this plot of the PSNR measurements of Table 5.1, it is clear that the iterative refinement quickly reaches its peak quality level and then stabilizes.

### 5.5.2 Processing Time Measurements

With regard to processing time, Table 5.2 lists the measurements to compute the disparity map of the Teddy scene for one side (i.e. left or right) only. The Teddy scene has a resolution of $450 \times 375$ and a disparity range of $[d_{min}, d_{max}] = [12, 53]$ (42 disparities), which we determined from its ground truth.

To complete the pipeline up to and including one iteration of the refinement (i.e. to compute $D_M^1$ or $D_M'^1$), our algorithm takes about 34 ms on an NVIDIA GeForce GTX TITAN Black. Of these 34 ms, about 29 ms (or 86%) is taken by the initial disparity estimation, whereas one refinement iteration only requires about 4.5 ms (or 13%). The remaining 1% (a negligible 0.5 ms) is taken to compute the local support windows. Adding a minimum of two more iterations of the refinement totals $34 + 2 \times 4.5 = 43$ ms, resulting in a very comfortable real-time performance at about 23 FPS (or about 163 MDE/s) to compute either $D_M^3$ or $D_M'^3$.

To arrive at those 23 FPS we have ignored one crucial detail however. In each new iteration of the refinement, the disparity cross-check requires both the left and right disparity maps of the previous iteration. More specifically, computing either $D_M^3$ or $D_M'^3$ requires both $D_M^2$ and $D_M'^2$ to be cross-checked against each other. This effectively halves the 23 FPS to 11.5 FPS, if we rely on a single GPU only. Fortunately, the disparity cross-check is the only point in the entire pipeline at which information from both sides is required. The left and right disparity maps can be computed completely independent from each other on separate

| Modules | | GT 640 | | GTX TITAN | | TITAN Black | |
|---|---|---|---|---|---|---|---|
| Sec. | Name | ms | % | ms | % | ms | % |
| 5.2 | Local Support Windows | 2.905 | 1.08 | 0.510 | 1.16 | **0.388** | **1.15** |
| 5.3 | Initial Disparity Estimation | 217.825 | 81.04 | 37.464 | 84.90 | **29.024** | **86.27** |
| 5.3.1 | • Per-Pixel Matching Cost | 4.684 | 1.74 | 0.748 | 1.70 | 0.638 | 1.90 |
| 5.3.2 | • Cost Aggregation | 213.141 | 79.30 | 36.716 | 83.20 | 28.386 | 84.37 |
| 5.4 | Iterative Disparity Refinement | 48.053 | 17.88 | 6.155 | 13.94 | **4.231** | **12.58** |
| 5.4.1 | • Disparity Cross-Check | 0.100 | 0.04 | 0.014 | 0.03 | 0.012 | 0.04 |
| 5.4.2 | • Bitwise Fast Voting | 47.693 | 17.74 | 6.100 | 13.82 | 4.186 | 12.44 |
| 5.4.3 | • Invalid Disparity Handling | 0.067 | 0.03 | 0.010 | 0.02 | 0.009 | 0.03 |
| 5.4.4 | • Median Filter | 0.193 | 0.07 | 0.031 | 0.07 | 0.024 | 0.07 |
| | Total | 268.783 | 100.00 | 44.129 | 100.00 | **33.643** | 100.00 |
| | • FPS | 3.72 | | 22.66 | | 29.72 | |
| | • MDE/s | 26.365 | | 160.602 | | 210.640 | |

**Table 5.2:** Per-module breakdown of absolute (ms) and percentage-wise (%) processing time, measured on an NVIDIA GeForce GT 640 (Fermi architecture) [NVIDIA Corporation, 2009], GTX TITAN and GTX TITAN Black (both Kepler architecture) [NVIDIA Corporation, 2012] for the Middlebury Teddy scene of Figure 5.2. This scene has a resolution of $450 \times 375$ and a disparity range of $[d_{min}, d_{max}] = [12, 53]$ (42 disparities). In general, the times listed are to process one side only ($D_M^1$ or $D_M'^1$). The exception is the local support windows entry. This entry includes computing the local windows for both the left and right input images, because both local windows must be combined into one global window during the cost aggregation. The disparity selection (section 5.3.3) is inherent to the implementation of the cost aggregation and cannot be timed separately. Boldfaced numbers are referenced in the text.

GPUs, after which they can be exchanged to prepare for the next iteration. If both GPUs are connected to the same bus, the exchange can be carried out with negligible overhead in memory management, resulting in a minimal impact on the 23 FPS. Figure 5.13 breaks down the time required to compute both $D_M^3$ and $D_M'^3$ on one GPU by combining the relevant modules of Table 5.2 appropriately.

Finally, adding two more iterations totals $43 + 2 \times 4.5 = 52$ ms, which still comes down to 9.5 FPS on one GPU and 19 FPS (134.6 MDE/s) on two GPUs to compute $D_M^5$ and $D_M'^5$.

## 5.6 Conclusion

We presented two main contributions to local stereo matching for dense disparity estimation.

First, we developed a reliable matching cost aggregation method. It combines two edge-sensitive support windows that adapt their shape to the underlying color information in the input images. One window follows the horizontal edges in the image, the other the vertical edges. Together they forms the final aggregation window shape that rigorously follows all

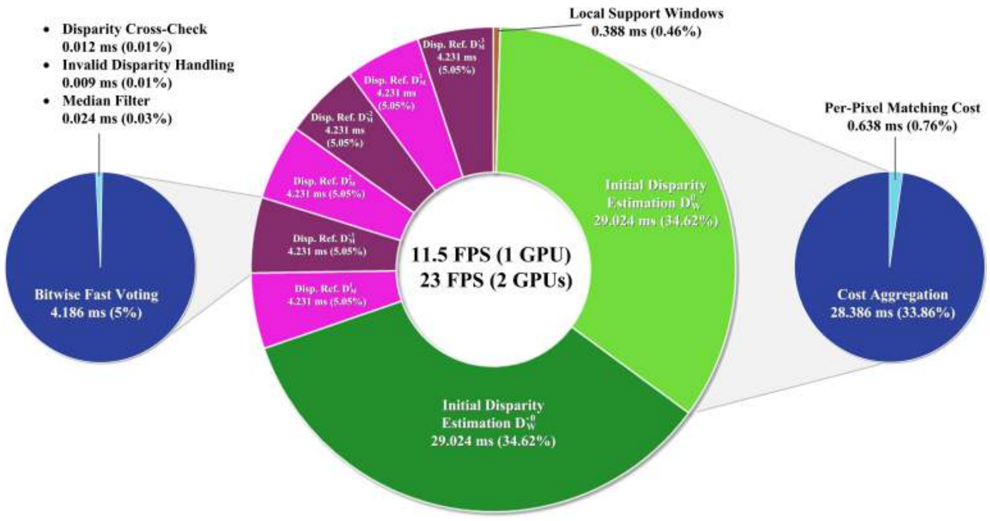


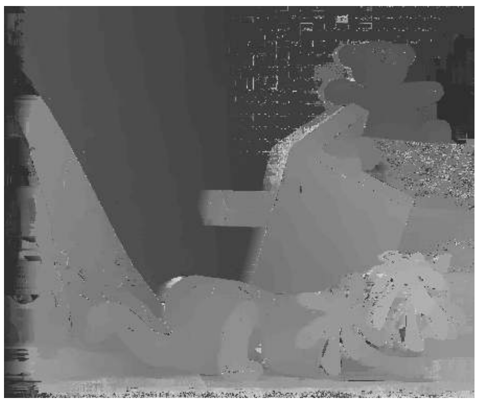
**Figure 5.13:** Detailed workload profiling to compute the disparity map up to and including three iterations of the refinement ($D^3_M$ and $D'^3_M$). Processing time is measured on an NVIDIA GeForce GTX TITAN Black graphics card for the 42 disparities of the $450 \times 375$ resolution Middlebury Teddy dataset.

object edges in varying directions. The windows cover image patches of similar color, which are assumed to belong to the same surface and therefore should possess the same disparity (or depth in the scene). Our shape-adaptive windows are represented by a single quadruplet per pixel, which renders the complexity of our solution comparable to existing methods.

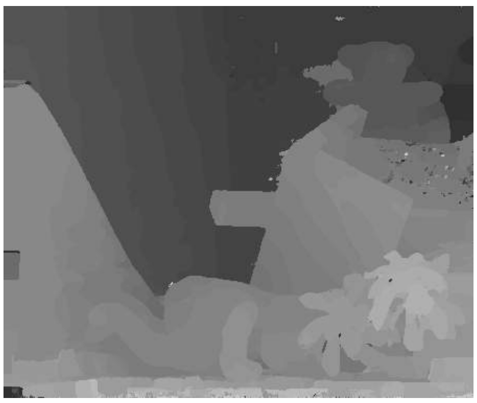
Second, we proposed a novel iterative disparity refinement process. It can be applied to any stereo matching algorithm and increases the quality of a disparity map with several dB PSNR. Its overall success is in large part attributable to the repeated interaction between four rigorously defined lightweight modules and especially between the disparity cross-check and the bitwise fast voting. Taking both the left and right disparity maps, the cross-check detects and removes all disparities that were incorrectly estimated, i.e. mainly occlusions around object edges and in the borders of the images. Next, by assuming that color discontinuity boundaries in the image are also depth discontinuity boundaries in the scene, the bitwise fast voting is able to reliably fill in the occlusions and smooth out disparities over patches of similar color. In between this, the invalid disparity handling helps to fill in invalid pixels that the bitwise fast voting cannot reach and the median filter removes speckle noise. It would be expected that an indefinite repetition would eventually have a detrimental effect on the quality of the disparity map. However, we observed that the interaction between the four modules prevents this from happening and instead the process tends to converge to a final solution.

Our whole algorithm has been designed for efficient GPU processing with negligible overhead, executes in real-time, is easy to understand and implement and generates smooth disparity maps with sharp object edges and little to no artifacts.

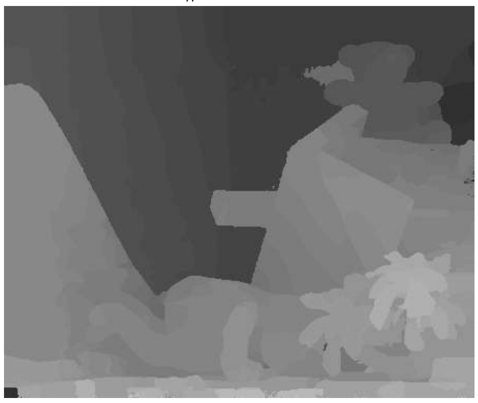
$I$



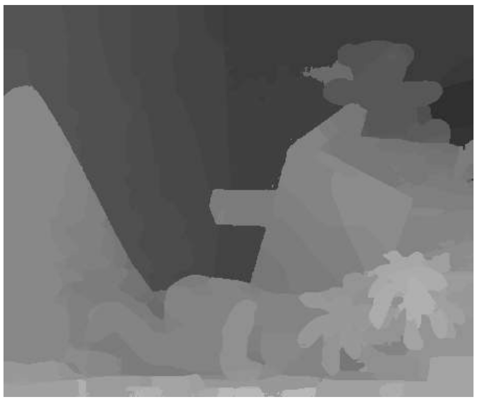
$D_W^0$ (19.40 dB)



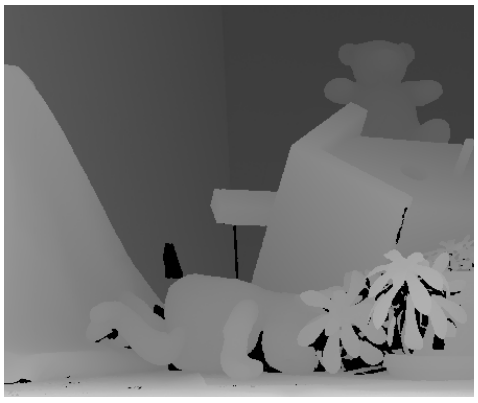
$D_M^1$ (27.53 dB)



$D_M^2$ (29.56 dB)



$D_M^5$ (29.96 dB)



$D_{GT}$

**Figure 5.14:** Our iterative refinement on the Middlebury Teddy scene: ($I$) left image, ($D_W^0$) initial disparity map, ($D_M^1$) – ($D_M^5$) 1 to 5 refinement iterations, ($D_{GT}$) ground truth.

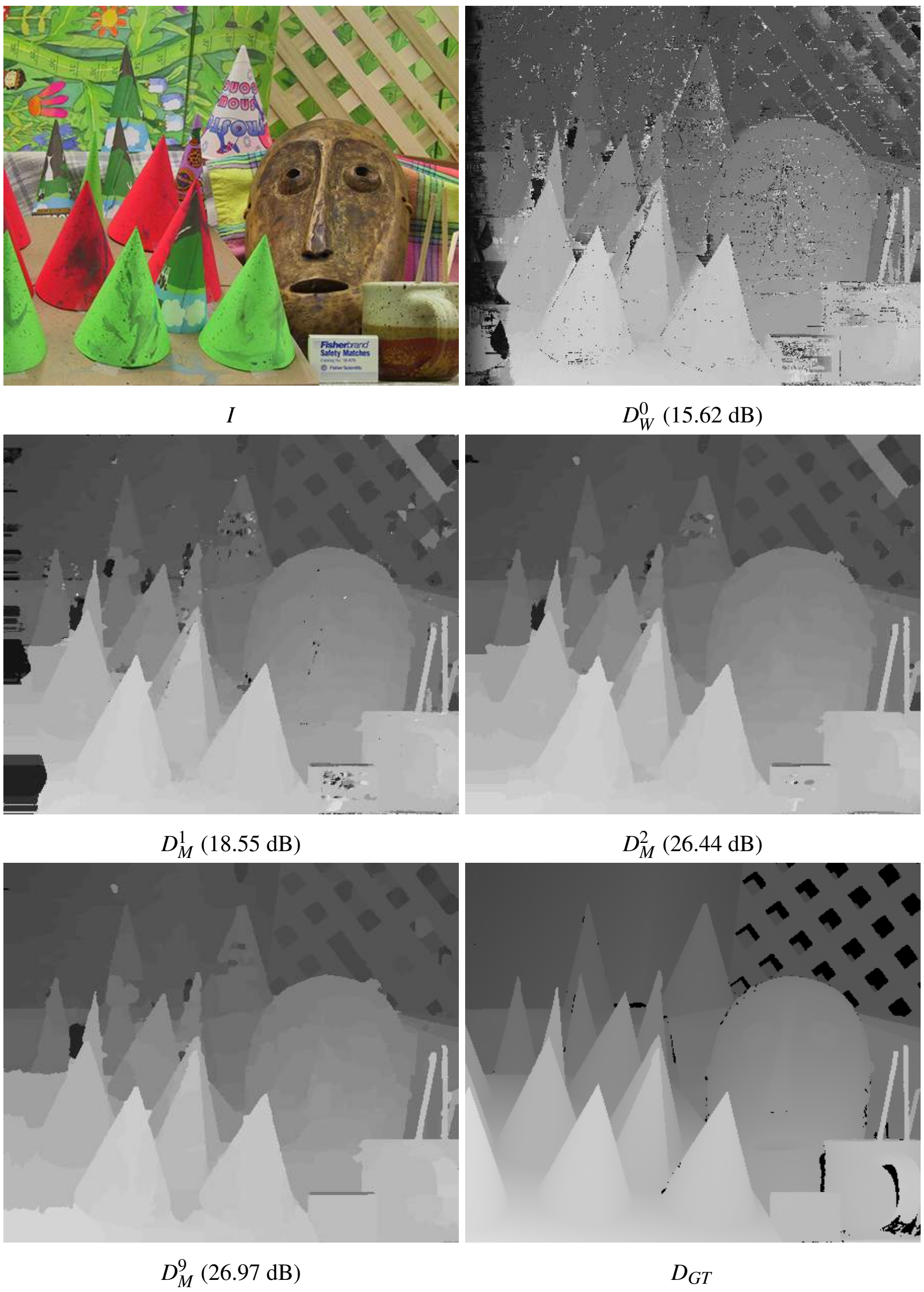


**Figure 5.15:** Our iterative refinement on the Middlebury Cones scene: ($I$) left image, ($D_W^0$) initial disparity map, ($D_M^1$) – ($D_M^9$) 1 to 9 refinement iterations, ($D_{GT}$) ground truth.

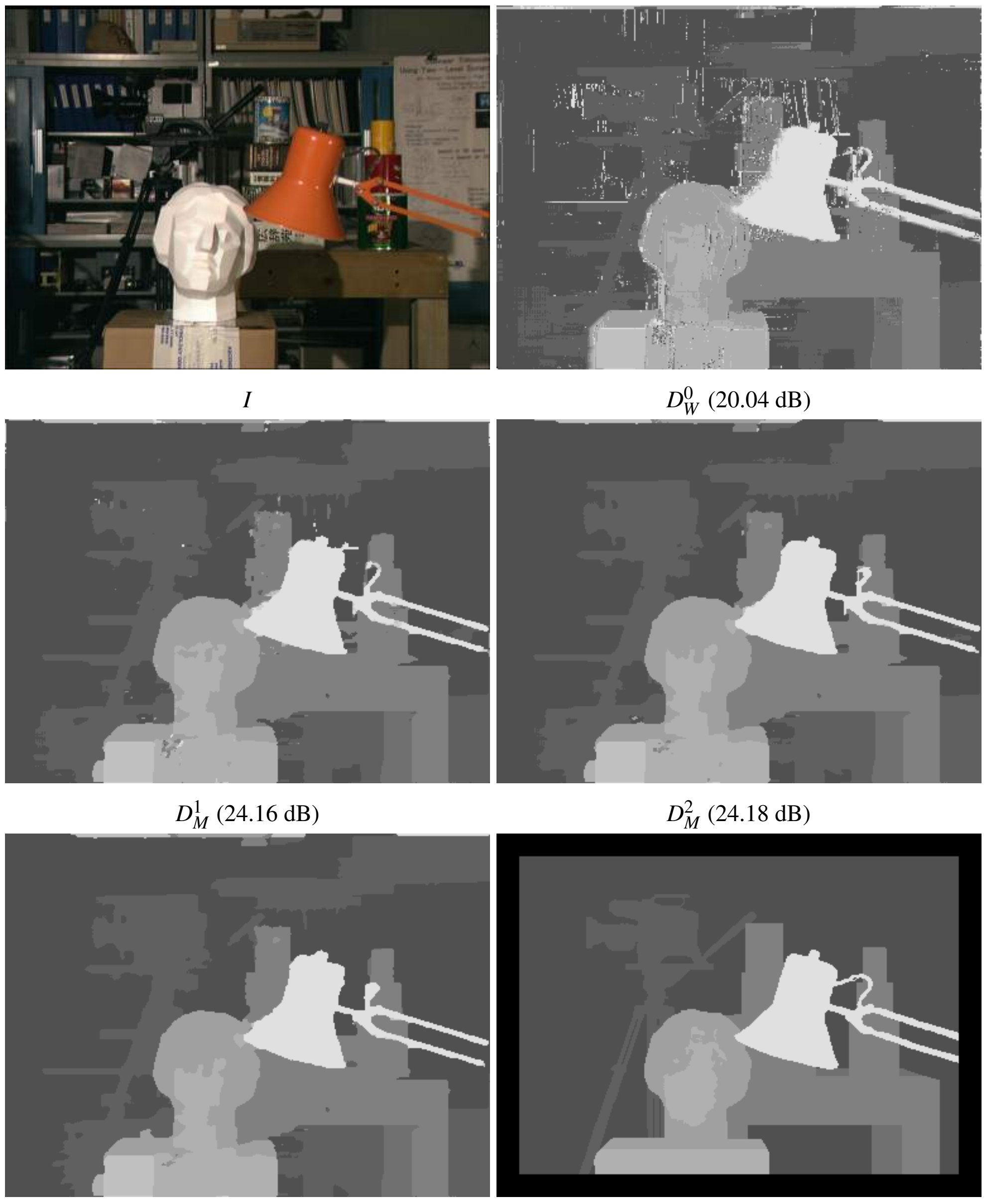


**Figure 5.16:** Our iterative refinement on the Middlebury Tsukuba scene: ($I$) left image, ($D_W^0$) initial disparity map, ($D_M^1$) – ($D_M^3$) 1 to 3 refinement iterations, ($D_{GT}$) ground truth.

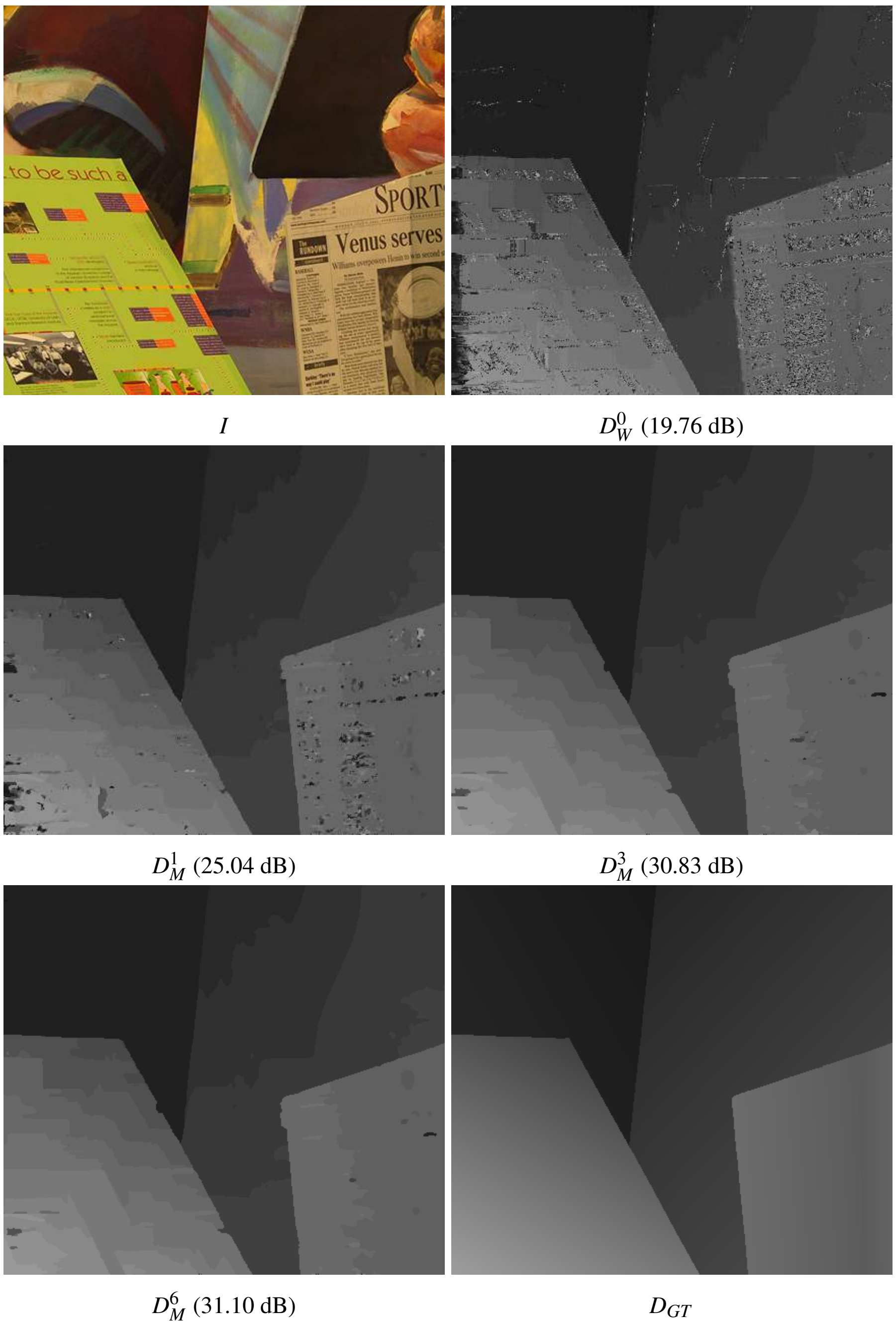


**Figure 5.17:** Our iterative refinement on the Middlebury Venus scene: ($I$) left image, ($D_W^0$) initial disparity map, ($D_M^1$) – ($D_M^6$) 1 to 6 refinement iterations, ($D_{GT}$) ground truth.

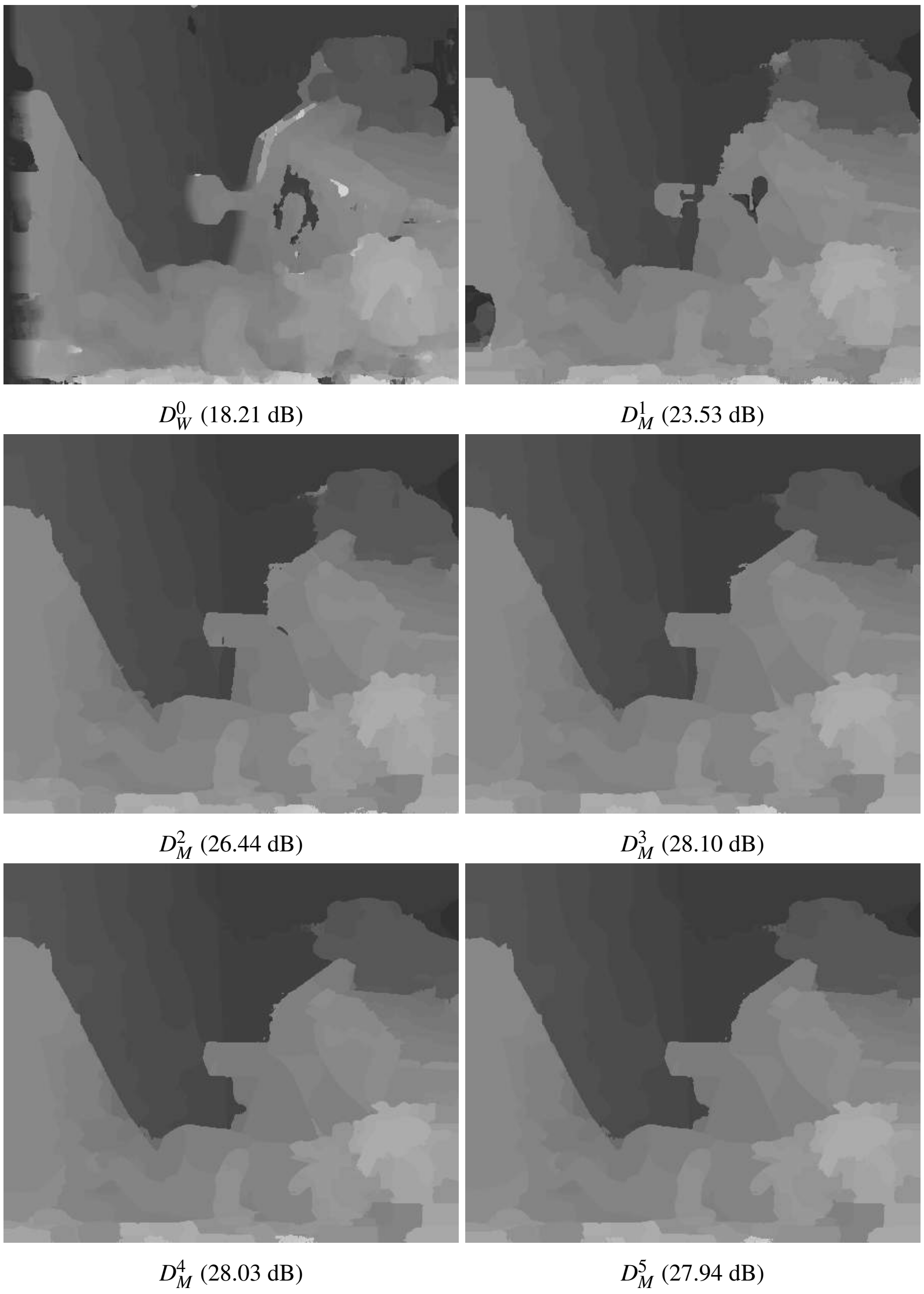


**Figure 5.18:** The Middlebury Teddy scene, but with ($D_W^0$) the initial disparity map computed using a conventional $17 \times 17$ square window and ($D_M^1$) – ($D_M^5$) subsequently refined using our iterative refinement. Compare with Figure 5.14.

### 5.6.1 Future Work

We currently consider the weakest link in our stereo matching algorithm to be the way the local support windows are determined. The pixel-wise color consistency check in section 5.2 is rather rudimentary. Relying on color-based image segmentation to more precisely define the local support windows has the potential to increase the matching quality considerably. Good segmentation to consider is mean-shift segmentation [Comaniciu and Meer, 2002; Gerrits and Bekaert, 2006] and superpixels [Zitnick and Kang, 2007]. Not only the cost aggregation stage, but also the iterative refinement (the bitwise fast voting counts bit votes within a coherently colored image patch) would benefit greatly from this.

As we will use our stereo matching algorithm to restore eye contact in video conferencing in chapter 6, it is especially worth considering incorporating temporal information [Davis et al., 2005]. By extending our shape-adaptive planar windows to volumetric grids in the temporal domain, we would be able to process over successive video frames.

# Chapter 6

# Stereo Interpolation

## Contents

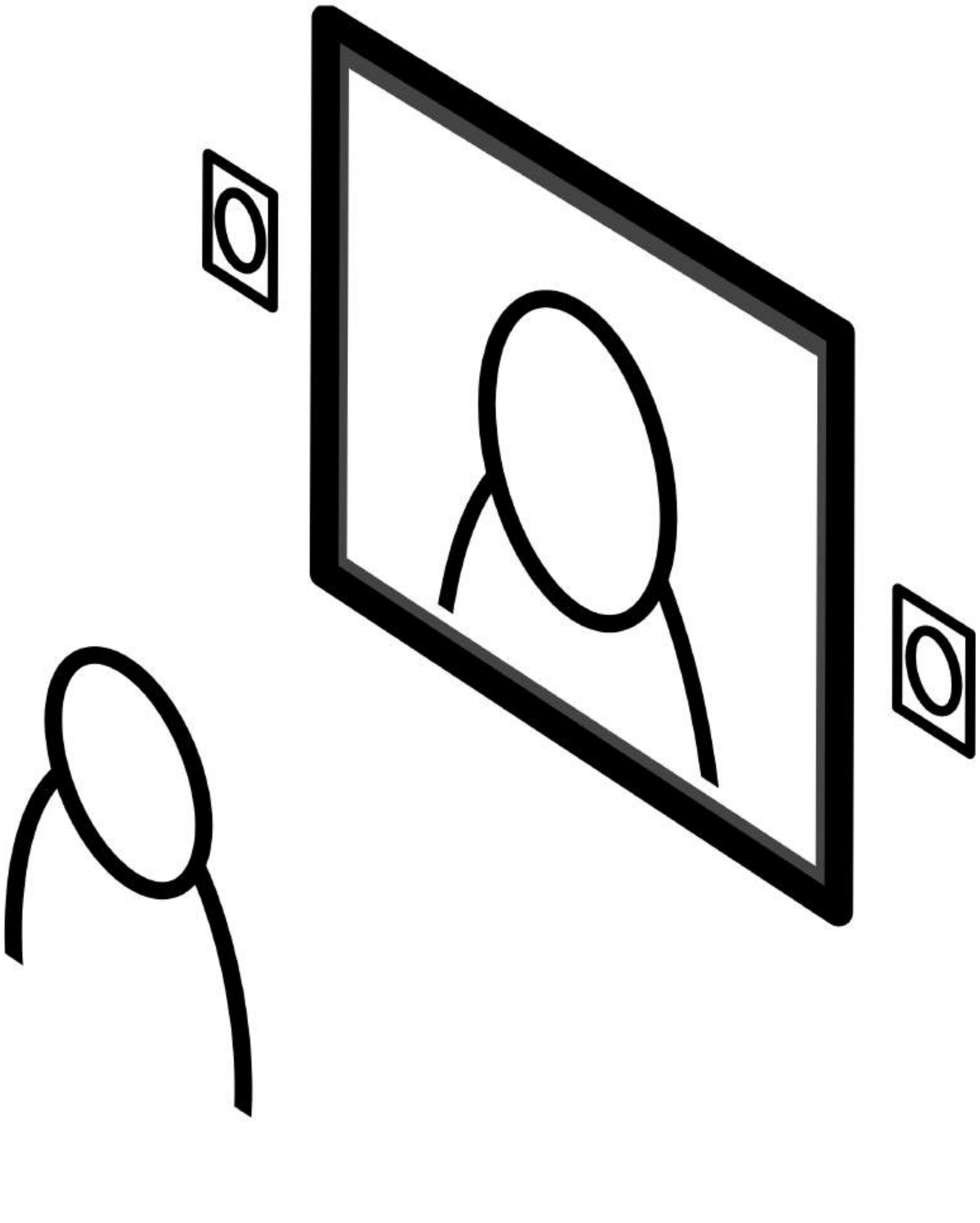


**Figure 6.1:** This chapter employs stereo interpolation to implement the two-camera solution depicted in the overview in Figure 1.1(b), p. 2. It relies on the stereo matching algorithm developed in chapter 5.

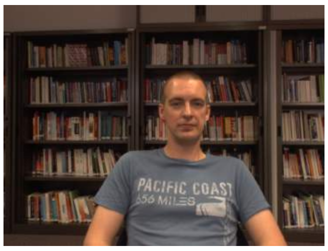
The environment mapping experiment in chapter 4 required only one input camera to restore eye contact, but offered very limited image quality, among other restrictions. In this chapter we therefore take a look at correcting eye gaze for one-to-one video conferencing by interpolating an image from two cameras positioned in a stereo configuration, as shown in the overview in Figure 1.1(b), p. 2, and repeated in Figure 6.1.

The same configuration is seen in Figure 6.2, which contains two camera views offset to the left (Figure 6.2(a)) and to the right (Figure 6.2(c)), capturing the user. These captured images will serve as the main dataset for the purpose of this chapter and our goal then becomes to reconstruct the unknown image of the virtual central camera (Figure 6.2(b)).

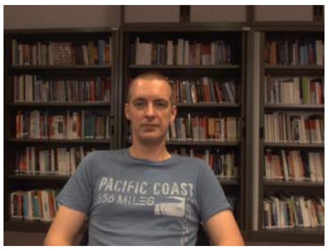
A big part of the road to that goal is our stereo matching algorithm that we developed in chapter 5. The stereo matching pipeline from Figure 5.3, p. 72, can be extended with a view synthesis part, shown in Figure 6.8, that enables us to reconstruct any viewpoint positioned anywhere on the baseline between the left and right viewpoints. Consequently, the reconstructed image will be eye gaze corrected if the correct intermediate position is chosen. To reconstruct an intermediate viewpoint, the view synthesis takes the rectified stereo images and *warps* them to the requested viewpoint, based on their disparity maps. In other words, to continue with our dataset, we require its disparity maps. They are shown in Figure 6.2(d) and Figure 6.2(f), as computed by our stereo matching algorithm. Additionally shown in Figure 6.2(g)–(i) are the disparity maps and interpolated color image as computed by the MPEG reference software (explained in section 6.4). Although of high quality, the MPEG algorithm is very computationally expensive and thus does not fit real-time applications such as ours. Nevertheless, we regard this as the ground truth solution to strive for.

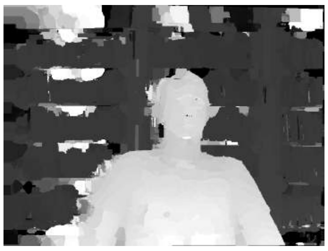
Stereo matching followed by view synthesis is well known as the unified depth-image-based rendering (DIBR) framework. We previously defined depth-image-based rendering as image-based rendering with implicitly determined geometry in section 2.1.2, p. 14. Also refer there for a conceptual explanation of stereo matching followed by view interpolation.

The depth-image-based-rendering algorithmic chain has been formalized by Scharstein [1996] and Rogmans et al. [2009c] and is depicted in Figure 6.4. This chain will be the leading thread throughout this chapter, but for our purposes we must adapt and extend it even further with the white modules. For starters, two cameras can rarely ever be manually set up in a perfectly rectified configuration and thus will have to be rectified before being offered as input to the stereo matching process. This is the responsibility of the rectification module in section 6.1. Next, in section 6.2, we introduce what we designate the complexity control module to increase the efficiency and quality of any stereo matching algorithm by limiting its disparity search range. The pipeline is finally completed to allow for novel view synthesis in section 6.3. Conventionally, the image warping module immediately warps the color images to the desired intermediate viewpoint. We, however, deviate from this by first warping the actual disparity maps to the desired viewpoint. Doing so allows us to further refine the intermediate disparity map (Figure 6.2(e)) in a more consistent manner, before composing the final color image in a postponed recoloring stage.

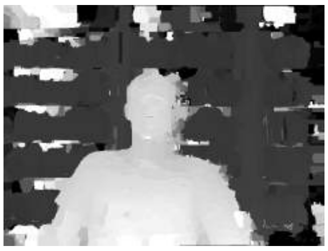

(a) Left input color image $I$.

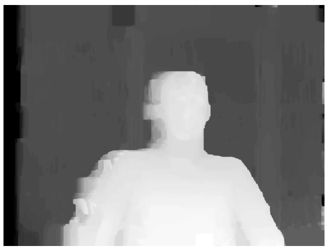
(b) Middle color image $\tilde{I}_\alpha$, to be reconstructed.

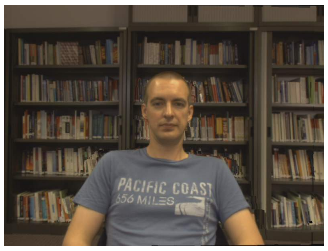

(c) Right input color image $I'$.


(d) Our left disparity map $D$.

(e) Middle disparity map $\tilde{D}_\alpha$, to be reconstructed.


(f) Our right disparity map $D'$.


(g) MPEG's left disparity map $D_{MPEG}$.


(h) MPEG's reconstructed middle color image $\tilde{I}_{MPEG}$.

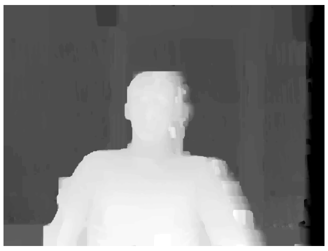

(i) MPEG's right disparity map $D'_{MPEG}$.

**Figure 6.2:** Our main dataset for the purpose of this chapter. (a), (c) Two cameras in a stereo configuration capture the user. (b) The goal of this chapter is to employ rectified stereo interpolation to reconstruct the image of the central camera, which consequently will have correct eye gaze. (d), (f) For this, we need the disparity maps as computed in chapter 5, (e) so that they can be warped by the view synthesis to the central viewpoint. (g), (i) The disparity maps and (h) reconstructed central color image computed by the MPEG reference software serve as ground truth solution.

In section 6.4 we compare our result with the MPEG reference algorithm. In section 6.5 we judge to what degree our stereo interpolation solution for eye gaze correction meets the requirements enumerated in the problem statement in section 1.1, p. 3. We conclude in section 6.6.

## 6.1 Stereo Rectification

Any stereo matching algorithm (including the one developed in chapter 5) assumes that its input images are rectified with respect to one another. What this means is that disparities between the images must be in the x-direction only, i.e. corresponding pixels must lie on the same image scanline, or more specifically the epipolar lines must run parallel with the X-axis.

The Middlebury dataset used in chapter 5 is inherently stereo rectified, but we may not assume that this is the case for the live imagery captured by the camera configuration of Figure 6.2. Put side by side in Figure 6.3(a) and Figure 6.3(b), it is indeed clear that these images are not rectified and therefore any attempt at scanline-based matching will fail. However, they can be efficiently rectified by resampling each according to their specific projective transformation. The process to compute these projective transformations (also called homographies) is well explained by Hartley and Zisserman [2004] and is summarized as:

(1) Find a set of point-to-point matches $p_i \leftrightarrow p'_i$ between the two images. At least 7 ($1 \leq i \leq 7$) correspondences are needed, though more are preferable for stability.

(2) From these correspondences, compute the fundamental matrix $F$ ($p_i F p'_i = 0$) and find the epipoles $e$ and $e'$ in the two images.

(3) Compute a homography $H'$ that maps the epipole $e'$ to the point at infinity, $H'e' = [1,0,0]^T$.

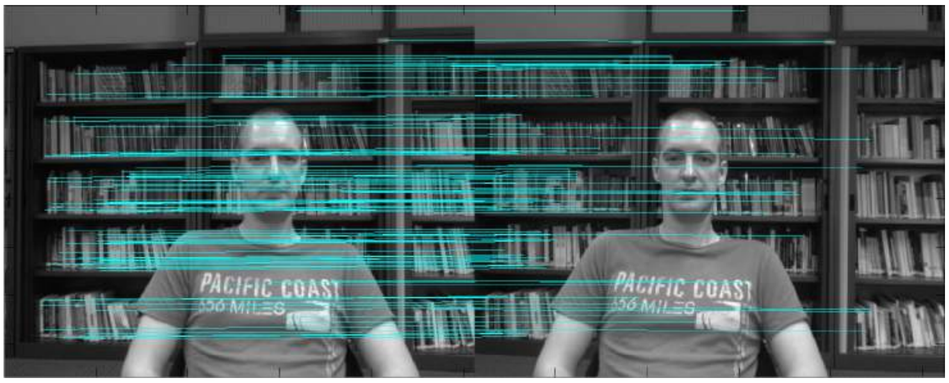
(4) Compute the matching homography $H$ that minimizes the least-squares distance $\sum_i d(Hp_i, H'p'_i)$.

(5) Resample the left image $I$ according to the homography $H$ and the right image $I'$ according to the homography $H'$.

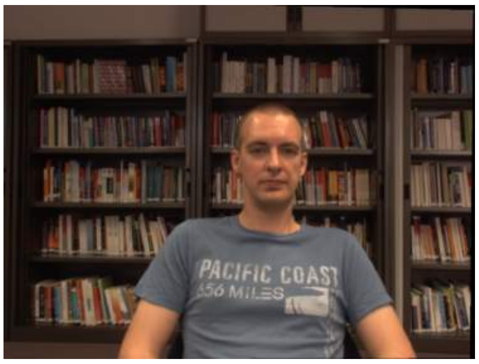
If the cameras are geometrically calibrated (as explained in section 2.4.1, p. 26), steps (1) and (2) can be skipped and the required point correspondences and epipoles can be computed directly from the known camera matrices [Hebert et al., 1995; Su and He, 2011]. Otherwise, for step (1) we detect and match SIFT (scale-invariant feature transform) [Lowe, 2004] features across the images, as shown in Figure 6.3(c). In any case, steps (1) to (4) only have to be performed once and thus the homographies $H$ and $H'$ can be computed offline as a preprocessing step, after which each newly captured frame during a live video conferencing session is rectified by the same precomputed homography. In other words, in step (5) the left input image $I$ is resampled using its homography $H$ according to the pseudo code:

(a) Captured left image.

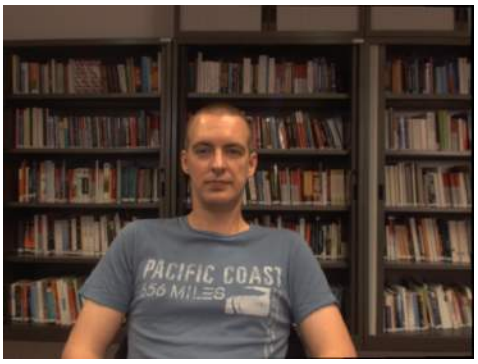
(b) Captured right image.



(c) SIFT correspondences between the non-rectified images.



(d) Rectified left image.



(e) Rectified right image.

**Figure 6.3:** Overview of the stereo rectification process. (a)–(b) First, the captured images have already been color calibrated (section 2.4.2) and corrected for lens distortion (section 2.4.3). (c) Then, SIFT correspondences are determined between the two images. (d)–(e) From these correspondences, two homographies are calculated that remap the pixels of the non-rectified images to their rectified counterparts. This is not a one-to-one pixel mapping, causing loss of information along the borders.

```
$J \leftarrow$ empty image
for all $p = (x_p, y_p)$ do
    $q \leftarrow H^{-1} p$
    $J(p) \leftarrow I(q)$                                        (6.1)
end for
$I \leftarrow J$
```

and completely analogous for the resampling of $I'$ using $H'$.

The resampling is done backwards: for each pixel in the rectified image $J$, its color is looked up in the non-rectified image $I$ by an inverse projection $H^{-1}$. Note also that this projection is not a one-to-one pixel mapping, which causes black patches of missing color information along the borders of the rectified images in Figure 6.3(d) and Figure 6.3(e). In practice these can be cropped before passing them on to the stereo matching algorithm, if so desired.

## 6.2 Complexity Control: Restricted Disparity Range

In section 6.2.1 we introduce a method that can greatly increase the quality of our stereo matching algorithm of chapter 5 – or of any other local stereo matching algorithm for that matter – and then go on to put that method to use as a control feedback loop to increase the performance of the depth-image-based rendering pipeline in section 6.2.2 [Rogmans et al., 2009a; Rogmans, 2013].

Keep in mind that we will further develop this idea in the context of plane sweeping for one-to-one and many-to-many communication in respectively section 7.2.5, p. 142, and section 8.3, p. 170.

### 6.2.1 Quality Increase

To achieve an increase in matching quality, the algorithm is executed twice, but over different disparity ranges. In the first pass, the entire disparity range $R = [d_{min}, d_{max}]$ is searched and the disparity map $D$ is computed as conventional. Next, an analysis on the disparity map $D$ is performed by computing its histogram $H(d)$. As illustrated in Figure 6.5, the histogram peaks indicate the disparities where objects are actually located. We can use this information to determine which subset $\bar{R} \subseteq R$ to take by comparing the histogram bin values with a given threshold $\delta$. A second pass then executes the matching algorithm over only the restricted disparity range $\bar{R}$. This efficiently filters out noise due to mismatches inherent to the full range $R$. The result is a higher quality disparity map $\bar{D}$.

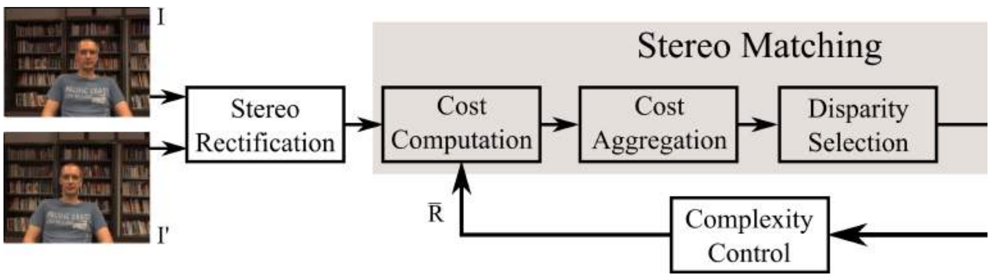


**Figure 6.4 (continued on facing page):** The depth-image-based rendering chain takes two rectified stereo images and reconstructs any intermediate viewpoint on their horizontal baseline. For the purpose of our eye gaze corrected video conferencing prototype, we extend this chain with an image rectification and complexity control module. We also adapt the view synthesis stage by adding a recoloring module.

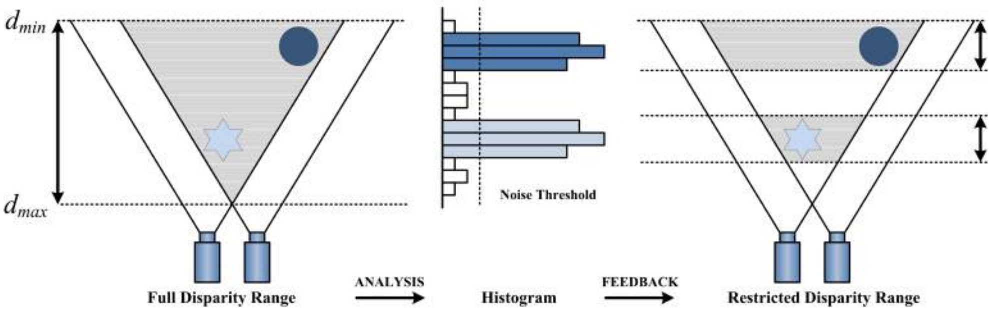


**Figure 6.5:** (left) First, the full disparity range is scanned. (middle) Peaks in the disparity map's histogram then indicate where objects are located in the scene. (right) This information is used to restrict the disparity range during a second scan.

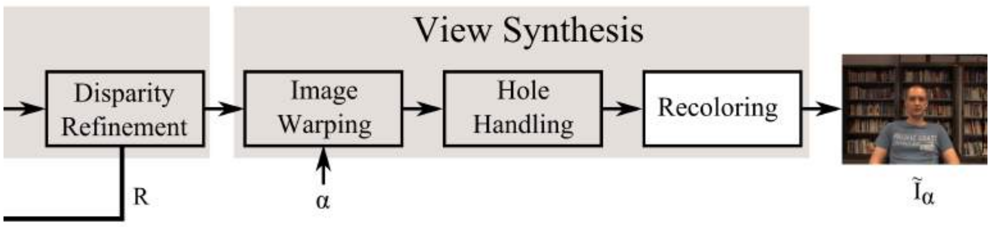
In other words, disparities for which less than δ pixels match, and which can thus be presumed to be noisy mismatches, are excluded from the search range in a second pass. Consequently, δ should be set proportionally to the image resolution:

$$\delta = \rho \times X_{res} \times Y_{res} \tag{6.2}$$

where we have observed in practice that $\rho = 0.006$ offers a good initial estimate.

Alternatively, instead of fixing the threshold δ, it can be made dynamic by setting it proportional to the discrete entropy of the histogram. Doing so will set the threshold low for complex scenes with fine-grained geometry and high when little geometry is present. Before applying the threshold, the histogram can also be smoothed to remove outliers.

In Figure 6.6 we apply this method to our dataset from Figure 6.2. This dataset has an SVGA ($800 \times 600$) resolution and its left viewpoint (after rectification) is repeated in Figure 6.6(a). In the first pass, our disparity estimation algorithm is executed over the full



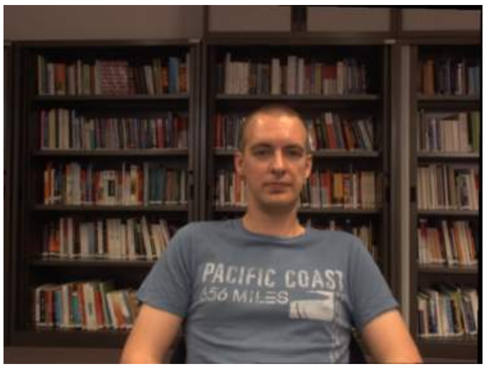
disparity range $R = [23, 106]$, which we determined from the MPEG reference algorithm that we regard as the ground truth (shown in Figure 6.2(g), repeated in Figure 6.6(b)). This results in the disparity map $D$ in Figure 6.2(d), repeated in Figure 6.6(d). Next, the histogram analysis restricts the disparity range $R$ to $\bar{R} = [38, 45] \cup [88, 103]$. This can be easily verified by visual inspection of the histogram $H(d)$ of the disparity map $D$ in Figure 6.6(c), where the threshold was set to approximately $\delta = 3000$ (the thin red line). A second pass of the algorithm over only the restricted disparity range $\bar{R}$ then results in a disparity map $\bar{D}$ of much higher quality, as shown in Figure 6.6(e). In the histogram, notice how the higher peak corresponds with the background (disparity range $[38, 45]$), while the lower peak represents the person in the foreground (disparity range $[88, 103]$). Everything in between will be cut out of the disparity range and will not be matched, but more importantly, will not be mismatched.

### 6.2.2 Complexity Decrease

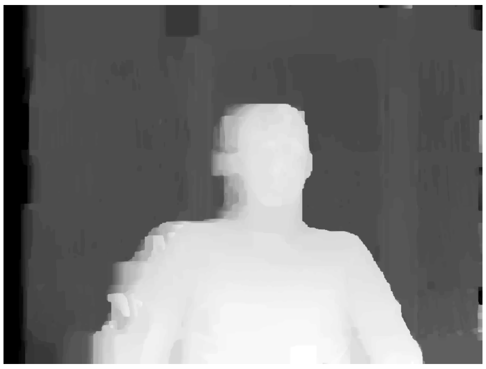
To enable the idea from section 6.2.1 as a control feedback loop for the DIBR pipeline depicted in Figure 6.4, suppose we first scan a lower resolution version of the input images over its full disparity range and analyze the resulting disparity map's histogram as before. The result of this analysis can then be used to restrict the disparity range of a higher resolution version of the same input images.

For the quarter-SVGA ($400 \times 300$) resolution version of our scene, the histogram analysis will restrict its full disparity range from $R_{1/4} = [11, 53]$ to $\bar{R}_{1/4} = [19, 23] \cup [44, 52]$, see Figure 6.7(a)–(b). We can use this result to more efficiently process the full SVGA ($800 \times 600$) resolution version of our scene, after correcting the disparity range $\bar{R}_{1/4}$ for the quadruple increase in resolution (doubled in both the horizontal and vertical direction) to $2 \times \bar{R}_{1/4} = [38, 46] \cup [88, 104]$. This very well approximates $\bar{R} = [38, 45] \cup [88, 103]$, the original restricted disparity range for the high-resolution images that we determined in section 6.2.1: $\bar{R} \approx 2 \times \bar{R}_{1/4}$.

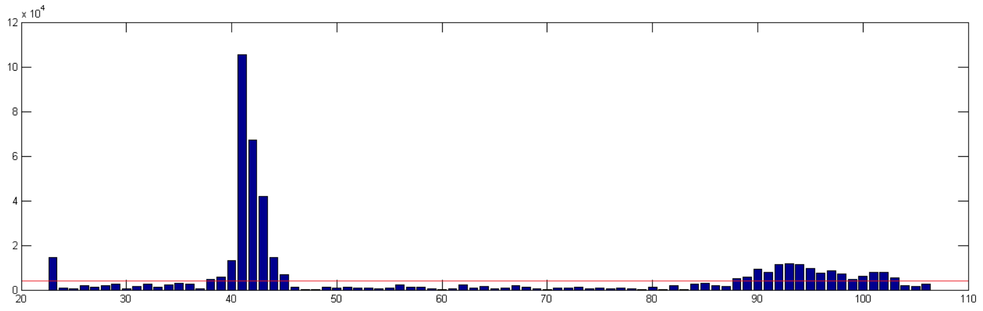
After the histogram analysis, the detected subset $2 \times \bar{R}_{1/4} = [38, 46] \cup [88, 104]$ is only 31% of the original high-resolution disparity range $R = [23, 106]$. As the low-resolution scan is at most 12.5% (1/4 image resolution and 1/2 disparity range) of the high-resolution complexity, a theoretical complexity reduction of 100% - (31% + 12.5%) = 56.5% can be



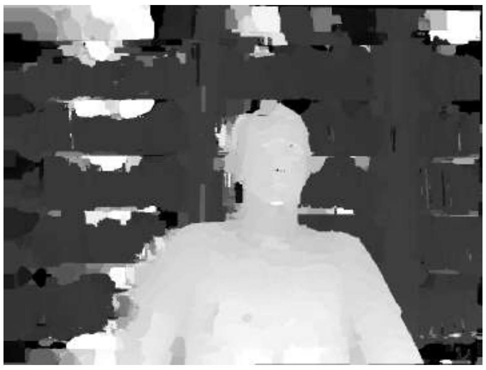
(a) Left input image $I$.



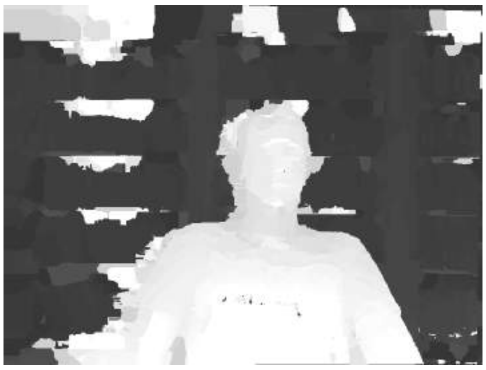
(b) Ground truth disparity map $D_{MPEG}$ of (a).



(c) Histogram $H(d)$ of (d), with threshold $\delta$ marked with a thin red line.

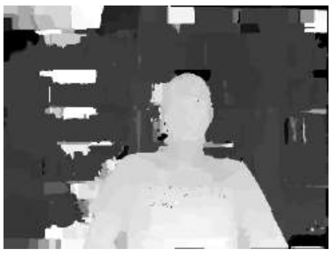


(d) Disparity map $D$ of (a), computed over the full range $R$.

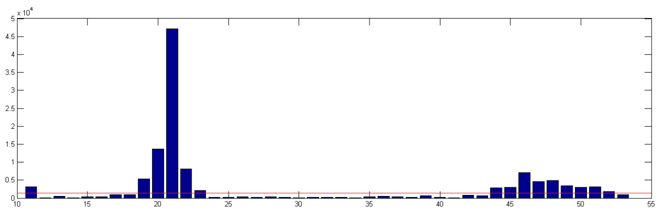


(e) Disparity map $\bar{D}$ of (a), computed over the restricted range $\bar{R}$, as determined in (c)

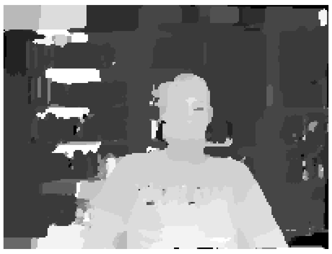
**Figure 6.6:** Overview of our histogram analysis scheme to improve the quality of the disparity map computed by any local stereo matching algorithm. (a) We use the left viewpoint of our dataset as input. (d) In the first pass, the matching algorithm is executed over the full disparity range $R$, (c) after which an analysis of the disparity map's histogram restricts the disparity range $R$ to $\bar{R}$ by applying a threshold $\delta$. (e) A second pass of the algorithm over only the restricted range $\bar{R}$ then results in a disparity map of much higher visual quality, (b) as can be noted by visual comparison with its ground truth.



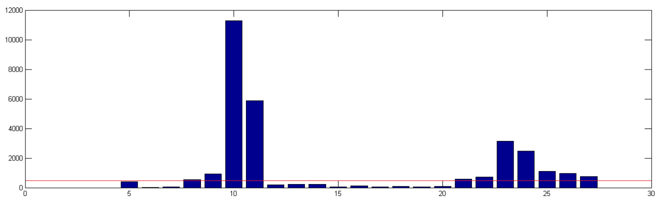
(a) Quarter-SVGA (400 × 300) disparity map.



(b) Histogram of (a).



(c) Sixteenth-SVGA (200 × 150) disparity map.



(d) Histogram of (c).

**Figure 6.7:** Histograms computed from lower resolution disparity maps can be used to restrict the disparity range for a higher resolution disparity map. This is possible because both lower resolution histograms still well represent the general shape of the full resolution histogram in Figure 6.6(c), if the stereo matching algorithm performs well.

achieved. In practice, however, at least some overhead comes into play. This is evident from the raw numbers listed in Table 6.1, as 1 - (36.764 ms + 142.203 ms) / 386.465 ms = 53.69%. Moreover, we did not take into account the time required to downsample the input images and to construct and analyze the histogram. Also note that the degree to which the disparity range can be restricted highly depends on the specific structure of the scene.

To investigate the extreme case, let us downsample the input images once more to a sixteenth-SVGA (200 × 150) resolution. This is illustrated by Figure 6.7(c)–(d). The histogram analysis now restricts the disparity range to $\bar{R}_{1/16} = [8,11] \cup [21,27]$. After correcting for the sixteen-times change in resolution, this becomes $4 \times \bar{R}_{1/16} = [32,44] \cup [84,108]$. This still quite well approximates $\bar{R} = [38,45] \cup [88,103]$ and comes down to a theoretical complexity reduction of 100% - (45% + 1.56%) = 53.44%.

The histogram information acquired from low-resolution disparity estimation over the full disparity range leads to accurate high-resolution disparity estimation over an adaptive range. The complexity of the histogram computation is relatively low and in turn the potential to accelerate the local stereo matching algorithm becomes very high. At the same time, matching quality is increased by excluding superfluous disparity hypotheses and thus inherently

| 1/16 SVGA (200 × 150) | | | 1/4 SVGA (400 × 300) | | | SVGA (800 × 600) | | |
|---|---|---|---|---|---|---|---|---|
| $R_{1/16}$ | #d | ms | $R_{1/4}$ | #d | ms | $R$ | #d | ms |
| $[5,27]$ | 23 | 5.032 | $[11,53]$ | 43 | **36.764** | $[23,106]$ | 84 | **386.465** |
| $\bar{R}_{1/16}$ | #d | ms | $\bar{R}_{1/4}$ | #d | ms | $\bar{R}$ | #d | ms |
| | | | $[19,23] \cup [44,52]$ | 16 | 15.166 | $[38,46] \cup [88,104]$ | 26 | **142.203** |
| $[8,11] \cup [21,27]$ | 11 | 2.847 | | | | $[32,44] \cup [84,108]$ | 38 | 192.032 |

**Table 6.1:** Disparity ranges for various resolutions of our dataset from Figure 6.2 and the time it takes (in ms) to compute a disparity map on an NVIDIA GeForce GTX TITAN Black (Kepler architecture) [NVIDIA Corporation, 2012]. The top row lists the full ranges, determined by the MPEG reference software. In the bottom rows we have restricted the disparity range of the sixteenth-SVGA (200 × 150) and quarter-SVGA (400 × 300) resolutions, using our histogram analysis scheme. As the resolution quadruples from the left to the right columns, so must the disparity ranges adapt by doubling their limits (and consequently, the number of processed disparities #d raises significantly). We can use this observation to increase matching quality while decreasing processing complexity. To this end, the boldfaced numbers are referenced in an example in the text for the SVGA (800 × 600) resolution.

preventing mismatches. In general, the less pronounced foreground fattening [Scharstein and Szeliski, 2002; Szeliski, 2010] an algorithm exhibits, the more the input image resolution can be reduced in the first pass, while still maintaining a representative histogram. In the case of our stereo matching algorithm from chapter 5, we have seen that the image dimensions can safely be reduced with a factor four. Considering that the complexity of the first pass is almost negligible, the complexity control add-on in Figure 6.4 allows for a speedup proportional to the amount of void space in the scanned volume. While the technique is applicable to all types of scenes, it proves to be particularly useful in video conferencing, as usually only the user and his background need to be scanned in a rather large space.

## 6.3 View Synthesis

The view synthesis part of the DIBR pipeline in Figure 6.4 takes two rectified stereo images as input, together with one or both disparity maps that were estimated by the stereo matching part, and reconstructs a novel intermediate view that is positioned anywhere on the horizontal baseline between the left and right input images. It accomplishes this in three steps, conceptualized in Figure 6.8 which can be regarded as a continuation of the stereo matching pipeline of Figure 5.3, p. 72. First, the image warping module (section 6.3.1) warps an input disparity map (and optionally its associated color image) to the desired viewpoint, after which the hole handling module (section 6.3.2) fills in the occlusions and holes that are inevitable due to the warping procedure not being a one-to-one pixel mapping. Lastly, the warped disparity map is recolored (section 6.3.3) to produce the final synthesized color image.

### 6.3.1 Image Warping

The image warping module accepts a color image and its joint disparity map and *forward warps* (as opposed to *inverse warping*, which can be regarded as a two-frame variant of the more generalized multi-frame plane sweeping algorithm) each pixel of the color image to the correct coordinate on the image plane of the intermediate viewpoint. The relative position of the intermediate viewpoint on the horizontal baseline between the left and right input viewpoints is determined by the parameter $\alpha \in [0,1]$, where by convention $\alpha = 0$ indicates the left viewpoint and $\alpha = 1$ the right. As such, we use $I_\alpha$ to denote the novel color image that results from a forward warping of the left input image $I = I_{\alpha=0}$. It is synthesized according to the following pseudo code [Scharstein, 1996; Seitz and Dyer, 1997; Rogmans et al., 2009c]:

```
for all $p = (x_p, y_p)$ do
    for $d \leftarrow 0$; $d \le \alpha \times d_{max}$; $d$++ do
        if $\alpha \times D(x_p + d, y_p) = d$ then
            $I_\alpha(p) \leftarrow I(x_p + d, y_p)$                    (6.3)
        end if
    end for
end for
```

where a pixel $p = (x_p, y_p)$ in the warped image $I_\alpha$ is filled with the color $I(x_p + d, y_p)$ from the source image $I$ whenever the disparity map $D$ indicates that the pixel at disparity $d$ maps to the intermediate position $\alpha$ according to $\alpha \times D(x_p + d, y_p) = d$. To save on computational complexity, only disparities up to $\alpha \times d_{max}$ are tested, as it is pointless to search for larger disparities than those that may occur by the rules of motion parallax.

Contrary to the algorithmic chain formalized by Scharstein [1996] and Rogmans et al. [2009c], we would like to postpone the synthesis of the actual color image to the very end of the pipeline (in section 6.3.3) and prefer to continue with the forward warped disparity map $D_\alpha$ instead, as this will allow us to make further refinements during the hole handling (in section 6.3.2). Thus the above warping algorithm is adjusted to:

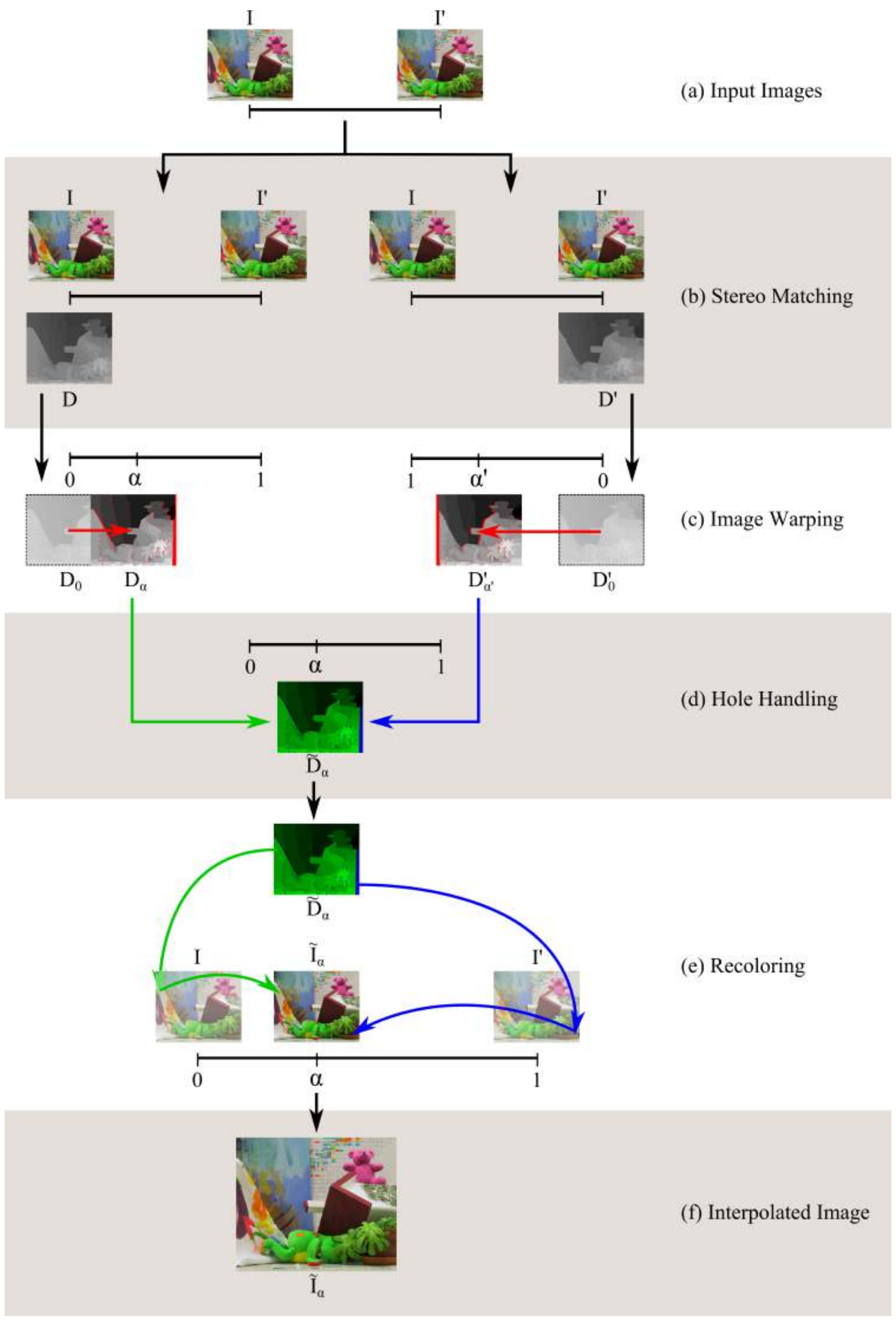
```
for all $p = (x_p, y_p)$ do
    $D_\alpha(p) \leftarrow INVALID$
    for $d \leftarrow 0$; $d \le \alpha \times d_{max}$; $d$++ do
        if $\alpha \times D(x_p + d, y_p) = d$ then
            $D_\alpha(p) \leftarrow \alpha \times D(x_p + d, y_p)$      (6.4)
        end if
    end for
end for
```

where the estimated disparity $d = \alpha \times D(x_p + d, y_p)$ for pixel $p$ is kept in $D_\alpha(p)$, with $D = D_{\alpha=0}$ (see Figure 6.8(c)(left)).

As the forward warp is not a one-to-one pixel mapping, it is possible for multiple pixels from the source image to map to the same pixel location in the novel image. However, by explicitly testing the candidate disparities from smaller to larger (in the for loop in Equation 6.3), a pixel is always filled with the color with the largest corresponding disparity, since any previous matches with smaller disparities are overwritten. This approach is known as the *occlusion compatible warping order*, because large disparities indicate foreground objects which occlude background objects with smaller disparities [McMillan, 1997]. Regardless, not all pixels in $I_\alpha$ (identically in $D_\alpha$ in Equation 6.4) will be filled in and the image will still contain gaps that are caused not only by occlusions, but also by mismatches and noise in the disparity map. To denote this variety of visual artifacts, we opt for the more generalized term *holes* [Scharstein, 1996; Rogmans et al., 2009c]. Holes are marked by setting some predefined invalid value.

To handle holes in section 6.3.2, we will also require the right disparity map $D'$ to be warped to the same intermediate position $\alpha$. This can be expressed from the standpoint of the right viewpoint by introducing the notation $\alpha' = 1 - \alpha$, such that the warped disparity map is denoted by $D'_{\alpha'}$, with now $D' = D'_{\alpha'=0}$ (see Figure 6.8(c)(right)). The warping of $D'$ to position $\alpha'$ then proceeds analogously to Equation 6.4:

**Figure 6.8** ***(facing page)*****:** Overview of our rectified stereo interpolation method. The goal is to reconstruct the image of any viewpoint located anywhere on the horizontal baseline between the left and right input images. The position of the intermediate viewpoint relative to the left viewpoint is determined by the baseline parameter $\alpha \in [0,1]$. Here, we have chosen $\alpha = 0.3$ (and thus $\alpha' = 1.0 - \alpha = 0.7$ relative to the right viewpoint). (a) The input is a pair of rectified stereo images $I$ and $I'$. In this figure we purposefully use the same Middlebury Teddy scene [Scharstein and Szeliski, 2003] that also served to develop our stereo matching algorithm in chapter 5. In fact, the whole Figure 5.3, p. 72, is contained in module (b), where our stereo matching algorithm estimates the disparity maps $D$ and $D'$ of the left and right input images respectively. (c) These disparity maps are then forward warped to the desired viewpoint by the image warping module (section 6.3.1). As the warping procedure is not a one-to-one pixel mapping, some pixels in the forward warped disparity maps $D_\alpha$ and $D'_{\alpha'}$ may not be filled in. These invalid pixels are marked in red. (d) Next, the hole handling module (section 6.3.2) combines both warped disparity maps into the final disparity map $\tilde{D}_\alpha$ of the intermediate viewpoint. Pixels are marked in green if taken from the left forward warped disparity map $D_\alpha$, whereas they are marked in blue if taken from the right forward warped disparity map $D'_{\alpha'}$. Because $\alpha = 0.3 < 0.5$, most pixels are taken from $D_\alpha$ and thus marked in green. Remaining pixels at occlusions around object edges and in the right image border are taken from $D'_{\alpha'}$ and thus marked in blue. (e) Finally, the recoloring module (section 6.3.3) recolors the intermediate disparity map $\tilde{D}_\alpha$ by taking colors appropriately from the left and right input images and thereby reconstructs (f) the desired interpolated image $\tilde{I}_\alpha$.


I
I'
(a) Input Images
I
I'
I
I'
(b) Stereo Matching
D
D'
0
α
1
1
α'
0
(c) Image Warping
$D_0$
$D_\alpha$
$D'_{\alpha'}$
$D'_0$
0
α
1
(d) Hole Handling
$\tilde{D}_\alpha$
$\tilde{D}_\alpha$
I
$\tilde{I}_\alpha$
I'
(e) Recoloring
0
α
1
(f) Interpolated Image
$\tilde{I}_\alpha$

```
$\alpha' \leftarrow 1 - \alpha$
**for all** $p = (x_p, y_p)$ **do**
    $D'_{\alpha'}(p) \leftarrow \mathit{INVALID}$
    **for** $d \leftarrow 0$; $d \leq \alpha' \times d_{max}$; $d{+}{+}$ **do**
        **if** $\alpha' \times D'(x_p - d, y_p) = d$ **then**                    (6.5)
            $D'_{\alpha'}(p) \leftarrow \alpha' \times D'(x_p - d, y_p)$
        **end if**
    **end for**
**end for**
```

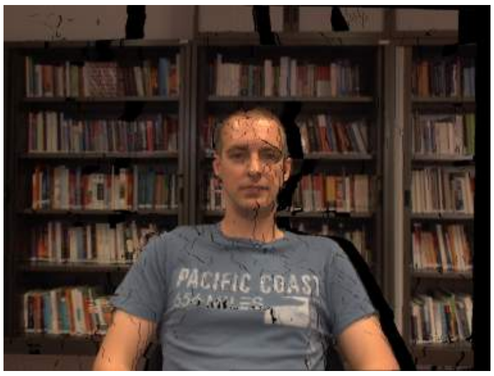
In Figure 6.9(a) the left image of our scene has been forward warped to $\alpha = 0.5$, while in Figure 6.9(b) the right image has been forward warped to the same relative position $\alpha' = 1 - \alpha = 0.5$. In Figure 6.9(a) occlusions from the left are clearly noticeable as black patches of missing color information, and likewise in Figure 6.9(b) for occlusions from the right. For this, we used the disparity maps that are the result of the iterative stereo matching process of chapter 5, combined with the disparity range restriction of section 6.2. They have also been forward warped in Figure 6.9(c) and Figure 6.9(d).

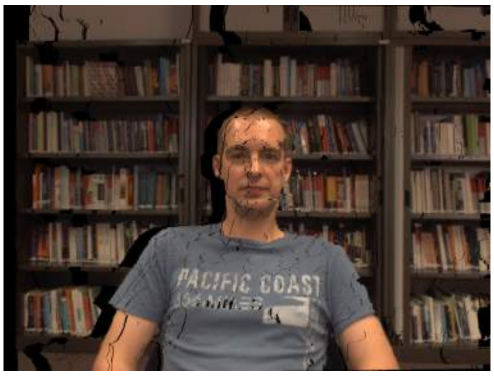
### 6.3.2 Hole Handling

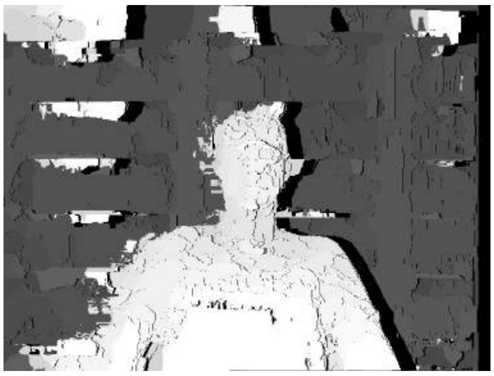
The two forward warped disparity maps $D_\alpha$ and $D'_{\alpha'}$ from section 6.3.1 must now be combined into one disparity map $\tilde{D}_\alpha$, as illustrated in the overview in Figure 6.8(d).

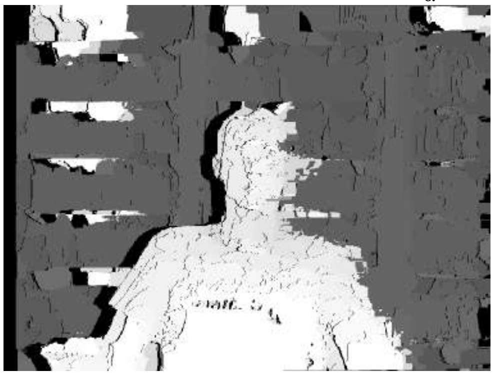
If $\alpha < 0.5$, it can be assumed that the occlusions in the left forward warped disparity map $D_\alpha$ will be less severe than the occlusions in the right forward warped disparity map $D'_{\alpha'}$, as the latter will have been warped over a distance of $\alpha' = 1 - \alpha > 0.5$. We therefore start from the left forward warped disparity map $D_\alpha$ and attempt to fill in any holes with information from the right forward warped disparity map $D'_{\alpha'}$ as follows:

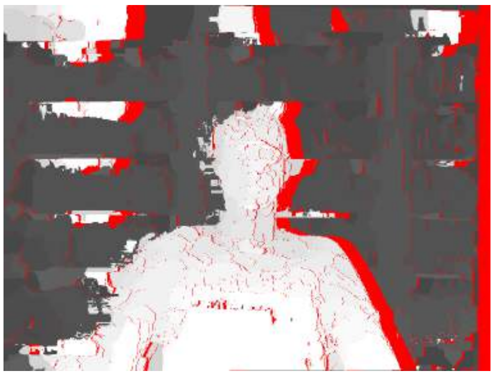
```
**for all** $p = (x_p, y_p)$ **do**
    **if** $D_\alpha(p) = \mathit{VALID}$ **then**
        $\tilde{D}_\alpha(p) \leftarrow D_\alpha(p)$
    **else if** $D'_{\alpha'}(p) = \mathit{VALID}$ **then**
        $\tilde{D}_\alpha(p) \leftarrow D'_{\alpha'}(p)$                    (6.6)
    **else**
        $\tilde{D}_\alpha(p) \leftarrow \mathit{INVALID}$
    **end if**
**end for**
```


(a) Left forward warped color image $I_\alpha$.


(b) Right forward warped color image $I'_{\alpha'}$.


(c) Left forward warped disparity map $D_\alpha$.


(d) Right forward warped disparity map $D'_{\alpha'}$.


(e) Occlusions in (c) marked in red.

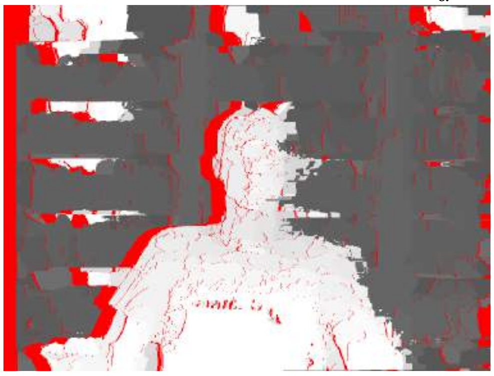

(f) Occlusions in (d) marked in red.

**Figure 6.9:** The left and right color images and disparity maps of our dataset from Figure 6.2 are forward warped to the same relative intermediate position $\alpha = \alpha' = 0.5$. Occlusions from the left (resp. right) are clearly noticeable as black patches, or marked in red in the bottom row.

Analogously, when $\alpha > 0.5$, the combined disparity map $\tilde{D}_\alpha$ is positioned closer on the baseline to the right viewpoint than it is to the left viewpoint, and thus we start from the right forward warped disparity map $D'_{\alpha'}$:

```
for all $p = (x_p, y_p)$ do
    if $D'_{\alpha'}(p) = VALID$ then
        $\tilde{D}_\alpha(p) \leftarrow D'_{\alpha'}(p)$
    else if $D_\alpha(p) = VALID$ then
        $\tilde{D}_\alpha(p) \leftarrow D_\alpha(p)$                    (6.7)
    else
        $\tilde{D}_\alpha(p) \leftarrow INVALID$
    end if
end for
```

Of course, when $\alpha = 0.5$ either one of the two approaches can be chosen.

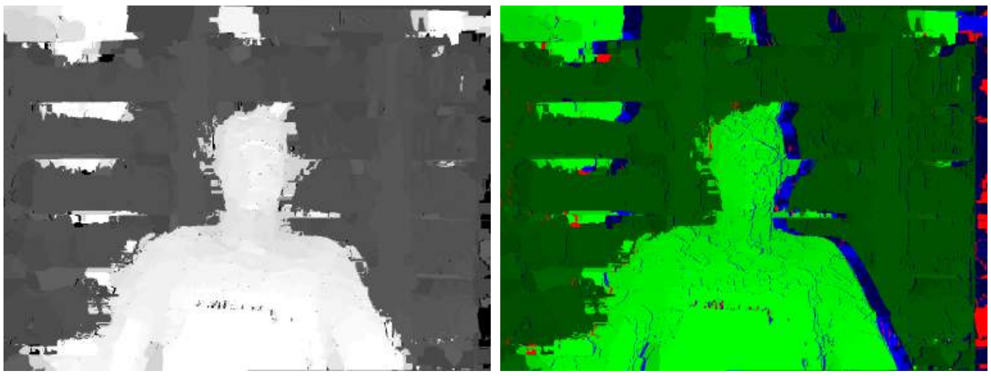
These routines will almost certainly leave some pixels in $\tilde{D}_\alpha$ marked as invalid, shown in Figure 6.10(a). How we proceed to handle these invalid pixels closely resembles the four-stage disparity refinement process of section 5.4, p. 81.

First, additional mismatches and occlusions can be detected by performing a cross-check on the forward warped disparity maps $D_\alpha$ and $D'_{\alpha'}$. If $D_\alpha(p)$ and $D'_{\alpha'}(p)$ are both valid, a pixel $p$ in $\tilde{D}_\alpha$ should still be marked as invalid if any of the following conditions are met:

$$\tilde{D}_\alpha(p) = \begin{cases} INVALID & \text{if } D_\alpha(p) \neq \alpha \times (D_\alpha(p) + D'_{\alpha'}(p)) \\ INVALID & \text{if } D'_{\alpha'}(p) \neq \alpha' \times (D_\alpha(p) + D'_{\alpha'}(p)) \\ VALID & \text{otherwise} \end{cases} \tag{6.8}$$

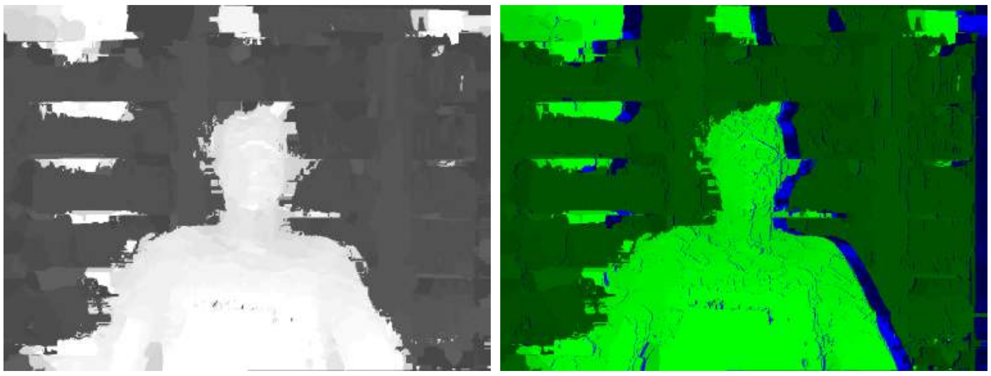
This cross-check can yield better results in heavily occluded scenes where the disparity refinement still failed. It is not unlike the disparity cross-check of section 5.4.1, p. 81. Second, any remaining invalid pixels are filled in using a scanline-restricted hole filling procedure that is completely analogous to the one of section 5.4.3, p. 85. Third, a median filter as the one of section 5.4.4, p. 86, can be applied. Special care should be taken, however, to differentiate between disparities that were warped from the left and ones that were warped from the right. Lastly, note that the bitwise fast voting of section 5.4.2, p. 81, cannot be applied, as we lack a reference color image to determine the local support windows.

The final hole handled disparity map of the intermediate viewpoint is shown in Figure 6.10(c). It remains to be recolored to obtain the synthesized color image.



(a) Occlusion handled disparity map $\tilde{D}_\alpha$, with some remaining invalid pixels.

(b) Same as (a), but color-coded.



(c) Any remaining invalid pixels from (a) filled in.

(d) Same as (c), but color-coded.

**Figure 6.10:** (a) The left and right forward warped disparity maps $D_\alpha$ and $D'_{\alpha'}$ from Figure 6.9 are combined into the occlusion handled disparity map $\tilde{D}_\alpha$ of the intermediate viewpoint. (b) Same as (a), but with pixels that were taken from the left marked in green, pixels from the right in blue, and remaining invalid pixels in red. (c) The final hole handled disparity map, where any remaining invalid pixels have been filled in. (d) Same as (c), but again color-coded. No red pixels remain.

### 6.3.3 Recoloring

In this final view synthesis step, the hole handled disparity map $\tilde{D}_\alpha$ (see Figure 6.10(c)) is recolored to produce the final synthesized color image $\tilde{I}_\alpha$ (see Figure 6.11(a)). The procedure is illustrated in the overview in Figure 6.8(e).

If a pixel $p$ of $\tilde{D}_\alpha$ took on a value of the left forward warped disparity map $D_\alpha$ (green in Figure 6.10(d)), its associated color must be fetched from the left color image $I$:

$$\tilde{I}_\alpha(p) \leftarrow I(x_p + \tilde{D}_\alpha(p), y_p) \tag{6.9}$$

If on the other hand the same pixel $p$ of $\tilde{D}_\alpha$ contains a value coming from the right forward warped disparity map $D'_{\alpha'}$ (blue in Figure 6.10(d)), its color must be retrieved from the right color image $I'$:

$$\tilde{I}_\alpha(p) \leftarrow I'(x_p - \tilde{D}_\alpha(p), y_p) \tag{6.10}$$

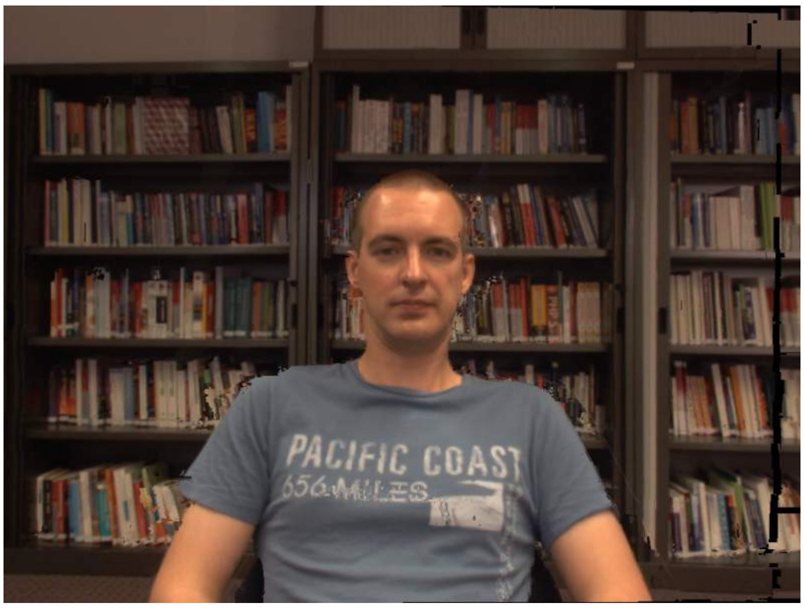
Note that because color information is taken from both the left $I$ and right $I'$ color images to put together the final output image $\tilde{I}_\alpha$, it is imperative that the images are photometrically calibrated according to the process outlined in section 2.4.2, p. 28.

## 6.4 Results

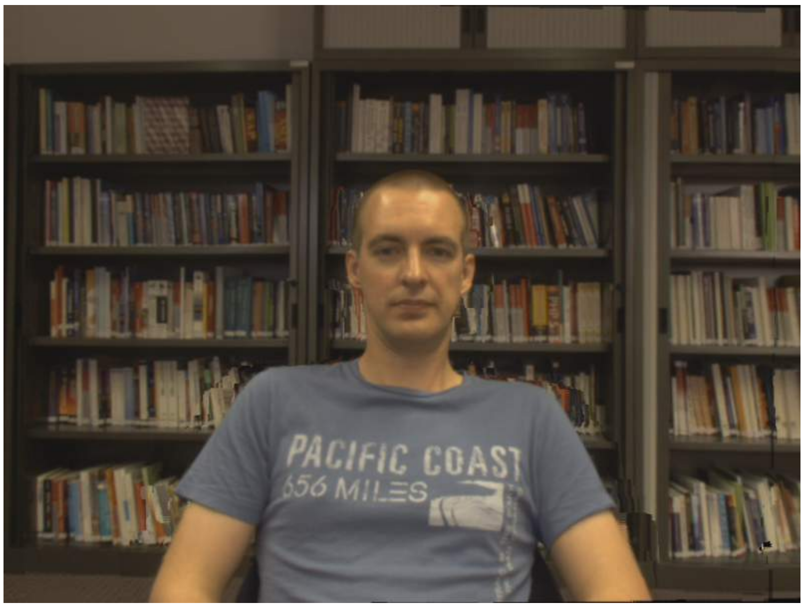
The final interpolated image $\tilde{I}_\alpha$ is shown in Figure 6.11(a). Figure 6.2(h), i.e. the image synthesized by the MPEG reference software, is enlarged in Figure 6.11(b) for easy visual comparison.

**About The MPEG Reference Software** The MPEG Reference Software for Depth Estimation (DERS) [Stankiewicz et al., 2013] and View Synthesis (VSRS) [Wegner et al., 2013] was originally developed by Nagoya University and Poznan University of Technology and updated with improvements throughout MPEG's standardization process. The depth estimation performs stereo matching with aggregation blocks and employs graph cuts for global refinement to compute depth maps for the input color images Boykov and Kolmogorov [2004]. The view synthesis then uses this depth information to warp and blend the color images to synthesize the desired viewpoint [Tanimoto et al., 2009]. Missing information in the synthesized image is filled with the closest available pixel values or by using texture patches when the invalid areas are relatively large [Ndjiki-Nya et al., 2011; Koppel et al., 2012]. The algorithm is described in more detail by Jorissen et al. [2014].

Visually, our result is completely on par with MPEG's result and both images very much resemble each other in every meaningful way. Both the foreground and the background of the interpolated image contain very few artifacts. Eye gaze is convincingly corrected and the eyes are clearly visible. A minimal amount of noise is still present, but the most noticeable artifact – one that both solutions suffer from – is the contour of visual disturbances closely surrounding the user. This *halo*-like effect is caused by occlusions during the disparity estimation that are then propagated by the forward warping. In this case, information that is missing in the left input image (marked in red in Figure 6.9(e)) must be filled in with information that is only available in the right image (marked in blue in Figure 6.10(d)). Although often challenging for pixel-wise local stereo matching algorithms, the results show that our algorithm is able to adequately handle this problem. It especially means that our iterative disparity refinement of chapter 5 is able to reliably fill in occlusions.



(a) Our interpolated image $\tilde{I}_\alpha$.



(b) MPEG's interpolated image $\tilde{I}_{MPEG}$.

**Figure 6.11:** (a) The final synthesized image $\tilde{I}_\alpha$ as the result of recoloring the final disparity map $\tilde{D}_\alpha$ of Figure 6.10(c). (b) The same viewpoint interpolated by the MPEG reference software, which we regard as the ground truth solution.

Computationally, the MPEG algorithm takes about half an hour to produce its result for the SVGA ($800 \times 600$) resolution. This includes both estimation of the scene depth and subsequent synthesis of the novel image, making it currently unviable for real-time applications. In contrast, Table 6.2 breaks down the time required by our algorithm for various resolutions. The view synthesis (i.e. section 6.3 in this chapter) is extremely lightweight, but relies heavily on accurate disparity estimation (i.e. all of chapter 5). The view synthesis, in fact, barely has any impact at all, as it requires a measly 0.115 ms for a QVGA ($320 \times 240$) resolution to still just under 12 ms for full high-definition ($1920 \times 1080$). Due to the weight of the disparity estimation, however, real-time performance of at least 15 FPS is obtained only for resolutions up to about $450 \times 375$ (common for Middlebury datasets). For VGA ($640 \times 480$) and SVGA ($800 \times 600$) resolutions, new frames still keep coming at a near real-time speed of 4.49 FPS and 2.57 FPS respectively, but starting from XGA ($1024 \times 768$) the frame rate drops – not unexpectedly – to 1 FPS and lower.

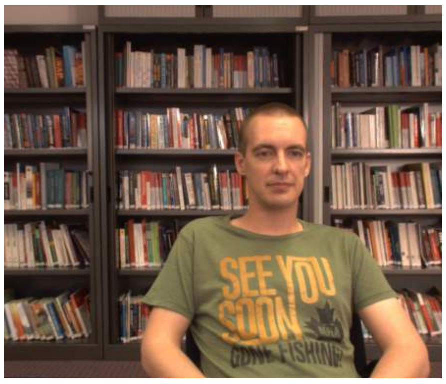
Two strategies are readily available to increase the frame rate. First, Table 6.2 lists the time required to match over the full disparity range. In section 6.2.2, however, we found that the algorithmic complexity can be decreased by about 50% by restricting the disparity range. From the discussion connected to Table 6.1, for example, we know that the time required for disparity estimation at the SVGA resolution can be brought back from about 386 ms to about 179 ms (36.764 ms + 142.203 ms). Considering the view synthesis is independent from the stereo matching and practically negligible in execution time (1.252 ms), this means a doubling of the frame rate from 2.57 FPS to 5.54 FPS. The same strategy can be applied to all resolutions. In addition, Table 6.2 also lists the time when executed on a single GPU. We saw in section 5.5.2, p. 89, that the disparity estimation time can once again by halved by splitting the algorithm over two GPUs and computing the left and right disparity maps separately. This would mean another doubling of the frame rate, thus successfully reaching about 10 FPS for the SVGA resolution.

Stereo matching (and consequently stereo interpolation) prefers small baselines. To investigate the wider-baseline case, we recorded the scene in Figure 6.12. As the baseline distance increases, the quality of the interpolated image usually drops off quickly. This is no different for the MPEG algorithm, which really struggles to interpolate the wider baseline in Figure 6.12(c). Occlusions around the head and arms are handled poorly and the text on the shirt exhibits ghosting artifacts. Our own algorithm in its standard form arguably does not perform much better, as many disturbing artifacts are visible in Figure 6.12(d). They are mainly caused by mismatches and are especially unacceptable on the face. To produce this result, however, we did not yet restrict the disparity range. This time our strategy is applied not necessarily to decrease the algorithmic complexity, but to increase the quality by preventing many mismatches, as explained in section 6.2.1. The final result in Figure 6.12(e) is very convincing. Some noise is present in the background where occlusions were filled in, but surprisingly few artifacts are left on the face and on the body.

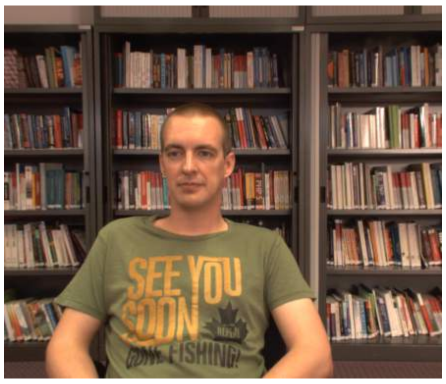
| Modules | | QVGA | 1/4 SVGA | Middlebury | VGA |
|---|---|---|---|---|---|
| Sec. | Name | (320 × 240) | (400 × 300) | (450 × 375) | (640 × 480) |
| 5 | Disparity Estimation | 21.964 | **36.764** | 62.718 | 221.675 |
| | • Range $R$ (#$d$) | [9,43] (35) | [11,53] (43) | [13,60] (48) | [18,85] (68) |
| | • FPS | 45.52 | 27.20 | 15.94 | 4.51 |
| | • MDE/s | 122.357 | 140.352 | 129.114 | 94.212 |
| 6.3 | View Synthesis | **0.115** | 0.205 | 0.307 | 0.666 |
| 6.3.1 | • Image Warping | 0.091 | 0.169 | 0.261 | 0.580 |
| 6.3.2 | • Hole Handling | 0.013 | 0.021 | 0.028 | 0.056 |
| 6.3.3 | • Recoloring | 0.011 | 0.015 | 0.018 | 0.030 |
| | Total | 22.079 | 36.969 | 63.025 | 222.341 |
| | • FPS | 45.29 | 27.04 | **15.86** | **4.49** |

| Modules | | SVGA | XGA | WXGA | FHD |
|---|---|---|---|---|---|
| Sec. | Name | (800 × 600) | (1024 × 768) | (1280 × 720) | (1920 × 1080) |
| 5 | Disparity Estimation | **386.465** | 965.295 | 2424.383 | 11452.515 |
| | • Range $R$ (#$d$) | [23,106] (84) | [30,135] (106) | [37,169] (133) | [56,254] (199) |
| | • FPS | 2.58 | 1.03 | 0.41 | 0.08 |
| | • MDE/s | 104.025 | 85.862 | 50.254 | 33.011 |
| 6.3 | View Synthesis | **1.252** | 2.473 | 3.617 | **11.485** |
| 6.3.1 | • Image Warping | 1.126 | 2.272 | 3.388 | 10.939 |
| 6.3.2 | • Hole Handling | 0.081 | 0.130 | 0.147 | 0.362 |
| 6.3.3 | • Recoloring | 0.045 | 0.071 | 0.082 | 0.184 |
| | Total | 387.717 | 967.768 | 2428.000 | 11464.000 |
| | • FPS | **2.57** | **1.03** | 0.41 | 0.08 |

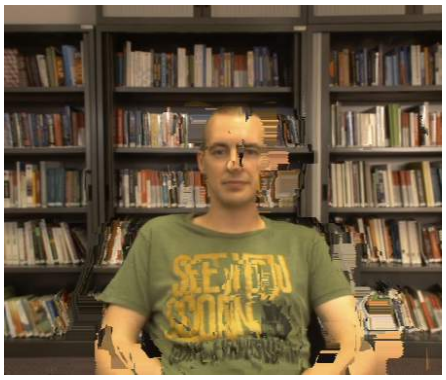
**Table 6.2:** Per-module breakdown of the processing time (in ms) and resulting frames per second (FPS) required to interpolate an eye gaze corrected image for our dataset of Figure 6.2 with resolutions ranging from Quarter VGA to Full HD. Time was measured on an NVIDIA GeForce GTX TITAN Black (Kepler architecture) [NVIDIA Corporation, 2012]. The view synthesis performs extremely efficiently for resolutions up to full HD and above, but relies heavily on accurate disparity estimation. Therefore, to get a more complete picture, the disparity estimation section has been included. It lists the time taken by our stereo matching algorithm to compute both the left and right disparity maps with one iteration of the refinement, i.e. both $D_M^1$ and $D_M'^1$ according to chapter 5. Also reported are the full disparity range and the FPS and MDE/s (million disparity estimations per second) achieved for each resolution. The time listed for the Middlebury (450 × 375) resolution is in line with the one reported in Table 5.2, Table 5.2, allowing us to evaluate our stereo matching algorithm for more resolutions. The performance can be further improved by restricting the disparity range (the 1/4 SVGA entry fits into Table 6.1) and by dividing the algorithm over two GPUs. Boldfaced numbers are referenced in the text.



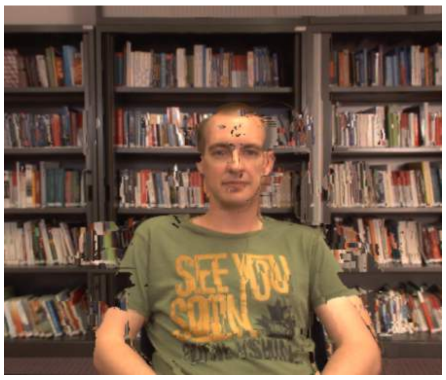
(a) Left input image.



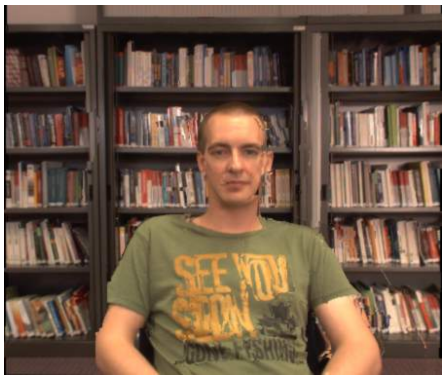
(b) Right input image.



(c) MPEG's result.



(d) Our result, without range restriction.



(e) Our result, with range restriction.

**Figure 6.12:** The small-baseline preference of rectified stereo forces us to either place the cameras around a narrower screen or assume a larger user-to-screen distance. (a)–(b) Input images with a wider baseline than the one in Figure 6.2. (c) Disturbing artifacts remain in MPEG's reference algorithm. Our own algorithm, however, performs much better (e) with disparity range restriction than (d) without.

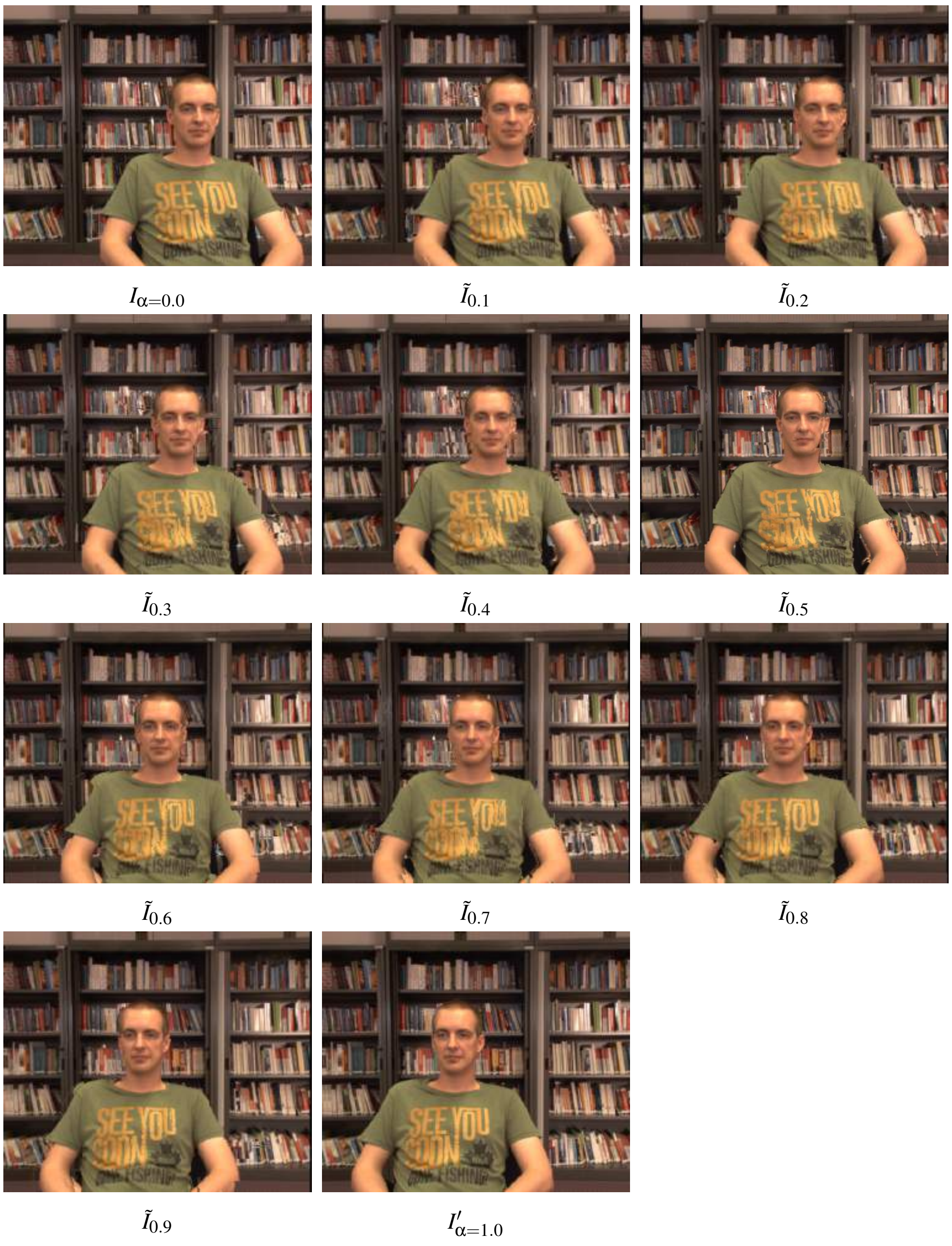


**Figure 6.13:** Visual quality is maintained as we sweep from the left image $I_{\alpha=0.0}$ to the right image $I'_{\alpha=1.0}$ by interpolating all images $\tilde{I}_{\alpha}, \alpha \in \{0.1, \ldots, 0.9\}$. The user is restricted to move in the horizontal space. However, as he is seated in the center, eye contact is lost in all viewpoints except $\tilde{I}_{0.5}$.

One last limitation that we run in to is the fact that rectified stereo only allows us to reconstruct viewpoints on its horizontal baseline. Consequently, the user is confined to move in the strict horizontal space between the cameras and his eye gaze is corrected only if his position corresponds to the position determined by the baseline parameter $\alpha$. This is evident from Figure 6.13, where we sweep from the left image $I_{\alpha=0.0}$ to the right image $I'_{\alpha=1.0}$ by interpolating a series of intermediate images $\tilde{I}_{\alpha}$. As the user is seated exactly in the middle, his eye gaze is only corrected when $\alpha = 0.5$. Deviating even slightly from this position will destroy eye contact. In practice, this imposes a not to be underestimated constraint on the usability of rectified stereo for natural eye gaze corrected video conferencing. Nevertheless, the visual quality is maintained for every interpolated viewpoint.

## 6.5 Requirements Evaluation

We are finally ready to evaluate our stereo interpolation solution for eye gaze correction. The evaluation is carried out on the requirements defined in section 1.1, p. 3. The scores correspond to the scale defined there and are plotted in Figure 6.14.

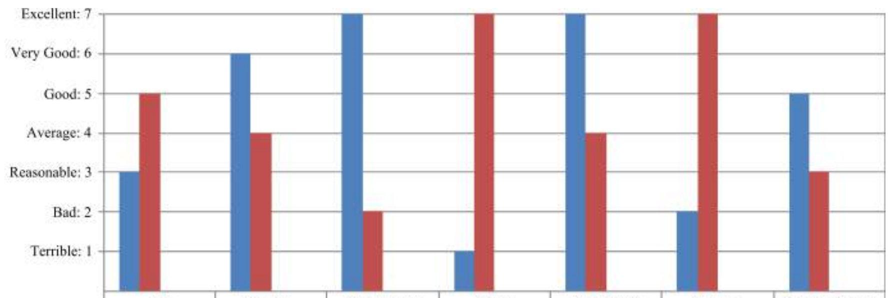
**Eye Contact** Eye gaze is corrected and eye contact is restored well. However, this is only the case as long as both users remain in the horizontally intermediate position determined by their respective baseline parameter $\alpha$. A trade-off with freedom of movement thus exists. **5/7 (good)**

**Spatial Context** It is difficult for stereo interpolation to offer more spatial context than an ordinary webcam can. Just as much of the user's background is visible, if interpolated correctly. Parallax effects are of course an advantage over an ordinary webcam that contribute to immersivity. **4/7 (average)**

**Freedom of Movement** The user's freedom to move is virtually non-existent. He is restricted to the position defined by $\alpha$, the relative position of the interpolated viewpoint on the horizontal baseline between the two cameras. Deviate even a little from this position and eye contact will be lost. **2/7 (bad)**

**Visual Quality** The few artifacts that remain in the interpolated image can nonetheless be distracting. They easily arise due to mismatching or occlusions and are notoriously difficult to prevent, especially if trying to achieve relatively high algorithmic performance. The eyes themselves, however, are very clearly discernible. Notwithstanding interpolation of pixel values and other image quality degrading operations performed by the image rectification, the synthesized image should be every bit as sharp as the input images. **7/7 (excellent)**

**Algorithmic Performance** The interpolation process itself is very lightweight, but depends heavily on accurate disparity estimation. Reliable stereo matching algorithms, in turn,



| | Eye Contact | Spatial Context | Freedom of Movement | Visual Quality | Algorithmic Performance | Physical Complexity | Communication Modes |
|---|---|---|---|---|---|---|---|
| 🟦 Environment Remapping | Reasonable | Very Good | Excellent | Terrible | Excellent | Bad | Good |
| 🟥 Stereo Interpolation | Good | Average | Bad | Excellent | Average | Excellent | Reasonable |
| 🟩 Plane Sweeping | | | | | | | |
| 🟪 Immersive Environment | | | | | | | |

**Figure 6.14:** Requirements evaluation of our stereo interpolation prototype for eye gaze correction. The scores correspond to the scale defined in section 1.1, p. 3. Environment remapping has been evaluated in section 4.7, p. 63. Missing data will be filled in as more solutions are developed in the next chapters.

are computationally expensive and have an impact on performance that is not to be underestimated. This is no different for our stereo matching algorithm. Although it is very performant compared with other local disparity estimation algorithms, it still performs much less efficiently compared with our environment remapping (chapter 4) and plane sweeping (chapter 7) approaches. **4/7 (average)**

**Physical Complexity** Modern cameras are small, cheap, and can easily be integrated into the bezel of any display. Moreover, they can be factory-calibrated to be in perfect stereo. **7/7 (excellent)**

**Communication Modes** Only one user can be active in front of the screen at the same time. **3/7 (reasonable)**

## 6.6 Conclusion

We took two cameras in a rectified stereo configuration and captured the user who is seated in the horizontal middle. We then interpolated an eye gaze corrected image by following the depth-image-based rendering pipeline, which essentially consists of a disparity estimation (chapter 5) and view synthesis (this chapter) stage. The view synthesis is very lightweight, but relies heavily on accurate disparity estimation.

To further improve the accuracy of the disparity estimation, we introduced the idea of limiting the disparity search range. In a two-pass process, the images are first matched over the full disparity range, but at a downsampled resolution. Peaks in the disparity map's histogram then indicate where objects are located in the scene and thus where the search range can be restricted during a second pass over the full image resolution. The result is a decrease in processing time, while at the same time an increase in matching quality. In general, the complexity reduction is proportional to the void space in the scene. While the technique is applicable to all types of scenes, it proves to be particularly useful in video conferencing, as usually only the user and his background need to be scanned in a comparatively large space.

The final eye gaze corrected image is virtually indistinguishable from the image interpolated by the (industry standard but not real-time) MPEG reference software. It is very sharp, contains few artifacts and the eyes are clearly discernible. A substantial amount of spatial context is also retained, in spite of most information that is not visible in both cameras being lost. This has the potential to offer convincing parallax effects if the user moves and the reconstructed viewpoint follows.

Nevertheless, we do run into a few limitations. First, the freedom to move is restricted to the horizontal baseline between the left and right capturing cameras. This causes eye contact to be difficult to maintain, as it depends on the user remaining stationary in the sweet spot of the reconstructed viewpoint. Finally, the small baseline preference of rectified stereo forces us to either place the cameras around a smaller screen or assume a larger user-to-screen distance. To overcome these limitations, we will turn to a technique that can handle multiple cameras with wider baselines in a more general (i.e. non-rectified) configuration. Coming up next, in chapter 7: plane sweeping.

### 6.6.1 Future Work

One angle that is worth investigating is further refining the forward warped disparity map during the hole handling stage. The formalization by Scharstein [1996] and Rogmans et al. [2009c] immediately warps the color images to the desired intermediate viewpoint, based on their associated disparity maps. We deviate from this by first warping the actual disparity maps to the desired viewpoint. We then fill in occlusions in the forward warped disparity maps using a straightforward nearest neighbor search on the scanlines, before composing the final color image in a postponed recoloring stage. More advanced occlusion handling could be looked at, among them (mostly global) methods based on segmentation [Bleyer and Gelautz, 2005; Wang and Zheng, 2008], graph cuts [Kang et al., 2001; Deng et al., 2007], belief propagation [Sun et al., 2005; Yang et al., 2009] and dynamic programming [Wang et al., 2006b]. A more local pixel-based solution would be to incorporate silhouette information by segmenting the user from his background, similar to what we will do for plane sweeping in chapter 7. This would allow to better identify and handle occlusions in the forward warped image, thereby to a certain extent alleviating the small-baseline limitation.

# Chapter 7

# Plane Sweeping

## Contents

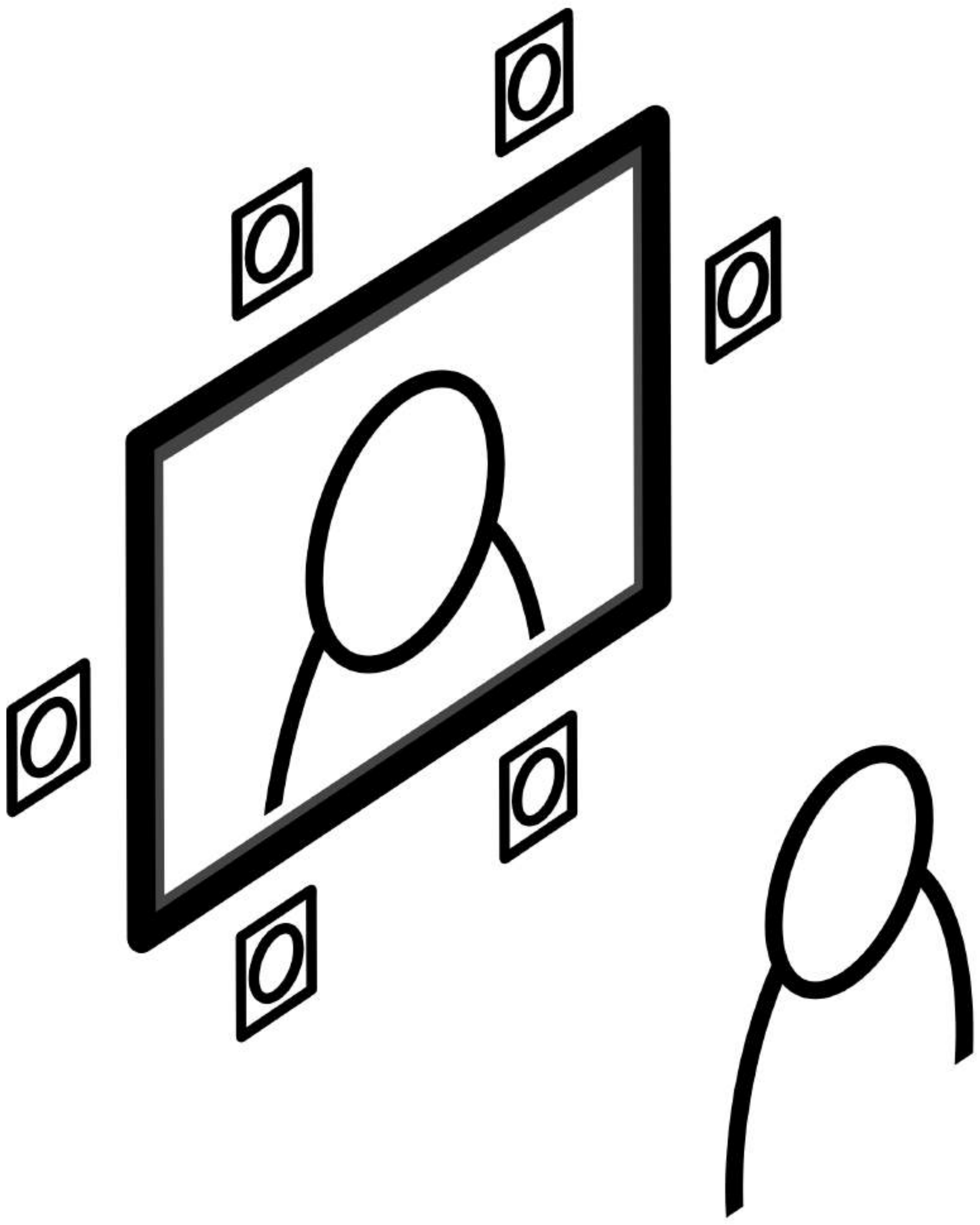


**Figure 7.1:** This chapter employs plane sweeping to implement the multi-camera solution depicted in the overview in Figure 1.1(c), p. 2.

In chapter 6 we corrected eye gaze by interpolating images captured by two cameras in a rectified stereo configuration. We ran into a number of constraints, the main one being very little freedom of movement, as the user is confined to the strict horizontal baseline. Furthermore, we saw that stereo matching is geared at smaller baselines, which forces the user to keep a considerable distance from the screen to avoid large occlusions. In this chapter we therefore aim to develop a fully functional end-to-end prototype for close-up one-to-one eye gaze corrected video conferencing [Dumont et al., 2008, 2009b; Rogmans, 2013; Dumont et al., 2014a]. We tackle the constraints of the previous chapter by mounting multiple – instead of only two – cameras around the screen. We will observe that solutions to typical problems for genuine practical usage emerge, such as continuous eye contact and a higher freedom of movement.

Figure 7.2 shows our prototype. It is the realization of its sketched conception in the overview in Figure 1.1(c), p. 2, repeated in Figure 7.1. The proposed six-fold camera setup is easily integrated in the monitor bezel and is used to interpolate an image as if a virtual camera captures that image through a transparent screen. However, this camera placement makes it impossible to enforce the condition of them being rectified for stereo matching. We therefore turn to plane sweeping [Yang et al., 2002], an image-based rendering technique that is able to interpolate cameras in a more general configuration. We conceptually explained plane sweeping and placed it in the context of depth-image-based rendering with implicitly determined geometry in section 2.1.2, p. 14.

Our stereo vision algorithms in chapters 5 & 6 were implemented in CUDA [Sanders and Kandrot, 2010; NVIDIA Corporation, 2007], which exposes the GPU directly as a massive pool of programmable parallel threads. In this chapter, however, we exploit the traditional graphics rendering pipeline for general-purpose computations by reprogramming its vertex and fragment processing stages through OpenGL and Cg [Mark et al., 2003; Rost et al., 2009; Shreiner, 2009]. This approach lends itself better to the inherent structure and scattered memory access patterns of plane sweeping [Rogmans et al., 2009b; Goorts et al., 2010], as we explained in section 2.3.1, p. 21.

This chapter is organized as follows. We begin with an in-depth overview of work related to plane sweeping in section 7.1. Next, section 7.2 describes our system's architecture in detail. To achieve real-time performance, we propose several carefully selected algorithms that are all appropriate for implementation on graphics hardware. Details on the implementation, together with a number of optimizations that map well to the traditional graphics pipeline, are given in section 7.3. The optimizations enable our prototype to achieve high subjective visual quality, while still allowing for further algorithmic advancement, without losing real-time capabilities. Results are presented in section 7.4, where we also present a fine-tuned set of user-independent parameters to optimize the application's end-to-end performance. Section 7.5 holds the by now familiar discussion on the requirements defined in section 1.1, p. 3, and section 7.6 ultimately concludes the chapter.

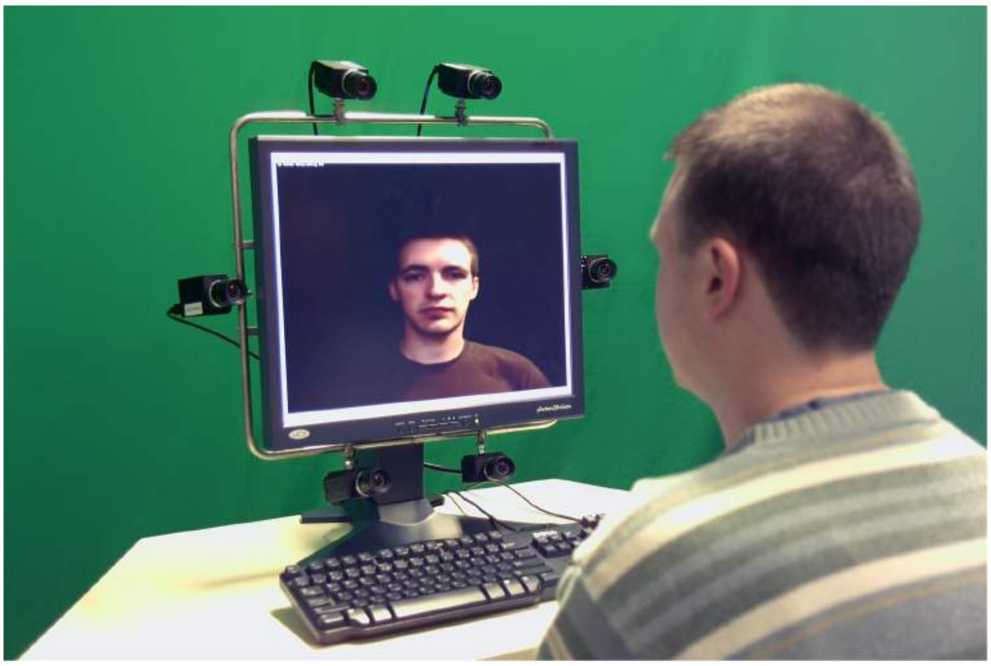


**Figure 7.2:** Our fully functional end-to-end prototype for close-up one-to-one eye gaze corrected video conferencing. Six cameras are mounted on a custom-made lightweight metal frame closely surrounding the screen: two above, two below and one to each side.

## 7.1 Related Work

Plane sweeping was originally developed by Collins [1996] as a true multi-image technique to simultaneously determine 2D feature correspondences and 3D positions of feature points in the scene. Yang et al. [2002] subsequently showed how to efficiently implement plane sweeping on commodity graphics hardware. They state that compared with other image-based rendering algorithms, their method easily achieves real-time performance, can deal with any arbitrary object shape and does not require silhouette information or complex geometric modeling of the scene.

Since then, many strategies have been devised to increase the accuracy of the plane sweep. Lots of research focuses on increasing the accuracy of the matching cost for each pixel on each plane, similar to the plethora of cost aggregation techniques for stereo matching that we discussed in section 5.1, p. 70. Yang et al. [2004b] show how to efficiently aggregate matching costs over variable square window sizes by exploiting the GPU's mipmapping functionality. Nozick et al. [2006] remove statistical outliers that deviate too much from the average of all colors that project to the same pixel on a plane. Geys et al. [2004] and Zach et al. [2008]

follow a more globally oriented approach to cost aggregation by incorporating graph cuts and Markov random fields respectively. Similarly, Kang et al. [2001] dynamically match a subset of neighboring viewpoints and explicitly label occluded pixels within a global energy minimization framework. Merrell et al. [2007] sweep from multiple viewpoints and fuse the results together while taking into account visibility constraints.

Other strategies include changing the shape or the direction of the planes to sweep. Zabulis et al. [2006] project on spherical surfaces instead of planes. Cornells and Van Gool [2005] take a two-pass approach whereby a depth map reconstructed in the first pass is swept again with small offsets to find local minima in the matching cost. Gallup et al. [2007] change the direction of the planes so that fronto-parallel surfaces are not favored. The sweep is performed for multiple orientations and the orientation that leads to the minimum data entropy of the depth map histogram is selected.

None of these strategies, however, look at the distribution of the planes in 3D space. We will do so in section 7.2.5 and section 8.3, p. 170.

Finally, plane sweeping serves in many other applications next to video conferencing. Among them are image stitching [Kang et al., 2004], interpolation of sports scenes [Goorts et al., 2014a; Goorts, 2014] and reconstruction of urban scenes [Pollefeys et al., 2008].

## 7.2 System Architecture

The core functionality of our system at one user's side is visualized in Figure 7.3. It consists of five consecutive processing modules, all of which run entirely on the GPU. Notice also the structural similarities between Figure 7.3 and the depth-image-based rendering pipeline in Figure 6.4, p. 106.

The input to our system is $N$ images $I_1, \ldots, I_i, \ldots, I_N$ that are captured from $N$ cameras $C_1, \ldots, C_i, \ldots, C_N$ mounted closely around the screen on a custom-made lightweight metal frame, as can be seen in Figure 7.2. To overcome occlusions we mount $N = 6$ cameras. Two cameras are placed above, two below and one to each side of the screen.

The preprocessing module (section 7.2.1) corrects for lens distortion and performs image segmentation. It is specifically designed to enhance both the quality and speed of the subsequent view interpolation and to ensure a high arithmetic intensity in the overall performance.

The view interpolation module (section 7.2.2) employs plane sweeping to interpolate an image $I_v$ as it would be captured by a virtual camera $C_v$ positioned behind the screen. The image is computed as if the virtual camera captured it through a completely transparent screen, thereby enabling the user to look directly into the virtual lens. The plane sweeping algorithm also inherently produces a joint depth map $Z_v$ with dense depth information of the scene.

The interpolated image will still contain a number of noticeable artifacts in the form of erroneous patches and speckle noise. The depth refinement module (section 7.2.3) is therefore specifically designed to tackle these problems under a real-time constraint. It marks

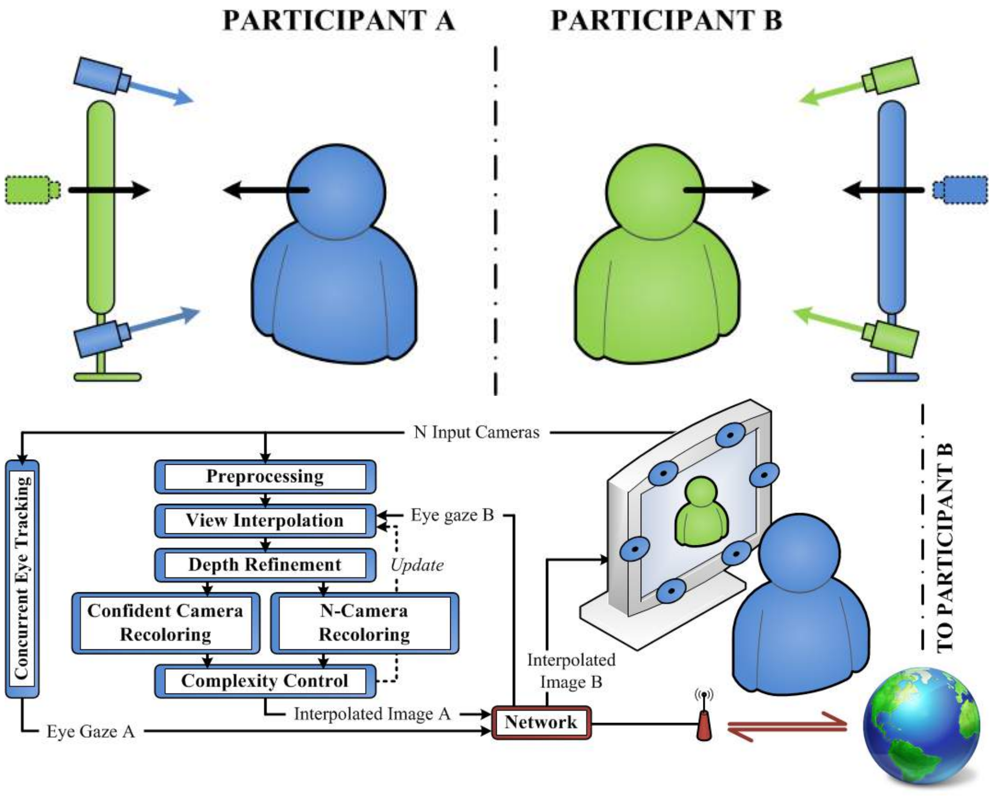


**Figure 7.3:** (bottom) System architecture of our end-to-end prototype. (top) An image is reconstructed as if a virtual camera (dotted lines) captures that image through a transparent screen. The virtual camera can be positioned freely, so that it always represents the remote user's eyes.

photometric outliers in the virtual image $I_v$ by detecting and correcting geometric outliers in the joint depth map $Z_v$. The result is the final refined depth map $\tilde{Z}_v$.

The depth refinement destroys the link between the interpolated image $I_v$ and the refined depth map $\tilde{Z}_v$. The recoloring module (section 7.2.4) restores this link by recoloring the refined depth map $\tilde{Z}_v$ to produce the final eye gaze corrected image $\tilde{I}_v$. Our system currently supports to recolor the synthesized image by either blending all $N$ cameras or by selecting the color from the camera which has the highest confidence of accurately capturing the required pixel.

In the final step, the refined depth map $\tilde{Z}_v$ is also analyzed to dynamically adjust the system and thereby avoid heavy constraints on the user's movements. The complexity control module for plane sweeping that we present in section 7.2.5 is the natural continuation of the one for stereo matching in section 6.2.

Next to the main processing on the GPU to synthesize the eye gaze corrected image, the virtual camera must also be correctly positioned to restore eye contact between the users. An eye tracking module (section 7.2.6) therefore concurrently runs on the CPU and determines the 3D position of the local user's eyes. The eye coordinates are then sent to the remote user's side to correctly conduct the virtual camera.

By sending the local eye coordinates to the remote user, the $N$ input images do not need to be sent over the network, but can be processed locally. This results in a minimum amount of data transfer, as only the eye coordinates and the synthesized eye gaze corrected image must be exchanged between both users.

### 7.2.1 Preprocessing

The first task of the preprocessing module is to correct the input images for lens distortion. We do this by pixel-wise applying the Brown-Conrady distortion model [Brown, 1966]. This model was explained in section 2.4.3, p. 29, with an example before and after correction for lens distortion shown in Figure 2.11, p. 29.

Next, each input image $I_i$ is segmented into a binary (foreground versus background) silhouette $S_i$. We propose two methods of segmentation: chroma keying and background subtraction. Both are evaluated pixel-wise and require very little processing power. Chroma keying on the green channel (green screening) is done according to:

$$S_i = \begin{cases} FG & \text{if } I_{i|g} > \rho_g \times \left(I_{i|r} + I_{i|g} + I_{i|b}\right) \\ BG & \text{otherwise} \end{cases} \tag{7.1}$$

where $I_{i|r}$, $I_{i|g}$ and $I_{i|b}$ are the red, green and blue channels of image $I_i$, $\rho_g$ determines the sensitivity of the green screen test, and $FG$ and $BG$ are predefined constants that indicate foreground and background. The pixel coordinate $p_i = (x_i, y_i)$ has been omitted for clarity.

Background subtraction serves in more practical scenarios:

$$S_i = \begin{cases} FG & \text{if } \|I_i - B_i\|^2 > \tau_f \\ & \text{or } \|I_i - B_i\|^2 \geq \tau_b \text{ and } \cos(\widehat{I_i B_i}) \leq \tau_a \\ BG & \text{if } \|I_i - B_i\|^2 < \tau_b \\ & \text{or } \|I_i - B_i\|^2 \leq \tau_f \text{ and } \cos(\widehat{I_i B_i}) > \tau_a \end{cases} \tag{7.2}$$

where $B_i$ is the background image for camera $C_i$, and $\tau_f$ (foreground), $\tau_b$ (background) and $\tau_a$ (angle) are experimentally determined thresholds that are subjected to parameter fine-tuning. The distance metric $\|I_i - B_i\|^2$ is the sum of squared differences (SSD):

$$\|I_i - B_i\|^2 = \sum_{c \in \{r,g,b\}} \left(I_{i|c} - B_{i|c}\right)^2 \tag{7.3}$$

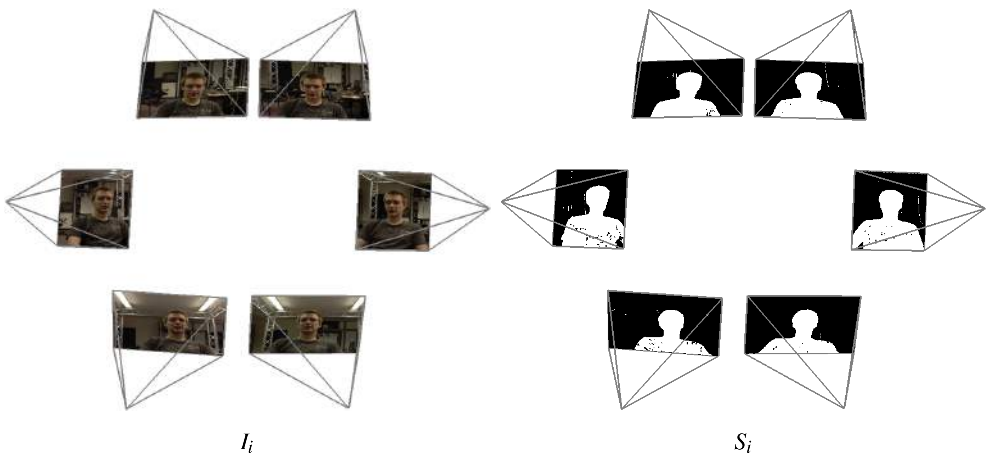


**Figure 7.4:** Background subtraction performed according to Equation 7.2 on our six input cameras. The geometric calibration of the cameras is known, which allows us to draw them here as they appear mounted around the screen. The remaining noise in the silhouettes will be removed by some morphological operations.

For shadow removal, the cosine of the angle $\widehat{I_i B_i}$ between the color component vectors of the image pixel and the background pixel is determined and thresholded against $\tau_a$. This renders the test robust against moderate illumination changes. Figure 7.4 illustrates background subtraction on our six input cameras.

The segmentation silhouettes will allow the upcoming view interpolation module to adequately lever speed and quality. Its performance may, however, still be profoundly impacted by any remaining noise in the silhouettes. As a final step, the noise is easily removed by a few morphological operations, i.e. erosions and dilations [Yang and Welch, 2002].

Lastly, we should note that in practice our system asks for the raw one-channel Bayer patterns (see section 2.4.4, p. 30) directly from the cameras and immediately transfers these to the GPU. It is then the very first task of the preprocessing module to convert them to three-channel RGB images. Doing so means a significant reduction in bandwidth consumption and thus improves performance considerably, as one-channel Bayer patterns are only a third the size of three-channel RGB images. Moreover, this avoids uncontrolled processing that would otherwise be integrated into the camera electronics. We will come back to this when we discuss specifics of the implementation in section 7.3.

### 7.2.2 View Interpolation

Figure 7.5(left) depicts how we interpolate the desired viewpoint by adopting and adapting the plane sweeping algorithm of Yang et al. [2002]. The 3D space is discretized into $M \geq 2$ planes parallel to the image plane of the virtual camera $C_v$. The planes have depths

$\{z_1, \ldots, z_j, \ldots, z_M\}$ relative to the virtual camera image plane and are uniformly distributed in the range $[z_{min}, z_{max}]$:

$$z_j = z_{min} + \frac{j-1}{M-1}(z_{max} - z_{min}) \tag{7.4}$$

so that $z_1 = z_{min}$ and $z_M = z_{max}$.

For every depth $z_j$, every pixel $p_v$ of the (to be reconstructed) virtual camera image $I_v$ is back-projected to a voxel $f_j$ on the plane with depth $z_j$ and subsequently reprojected to a pixel $p_i$ in each input camera image $I_i$:

$$f_j = M_v^{-1} \times K_v^{-1} \times T_j \times p_v \tag{7.5}$$

$$p_i = K_i \times M_i \times f_j \tag{7.6}$$

with $M = [R|t]$ and $K$ the extrinsic and intrinsic camera calibration matrices as defined in section 2.4.1, p. 26. $T_j$ is a transformation matrix that contains a translation and scaling to define the position and extent of the plane with depth $z_j$ in world space:

$$T_j = \begin{bmatrix} z_j \times X_{res} & & & 0 \\ & z_j \times Y_{res} & & 0 \\ & & 1 & z_j \\ & & & 1 \end{bmatrix} \tag{7.7}$$

with $X_{res} \times Y_{res}$ the resolution of the virtual camera image. The relation between all these coordinate frames is clarified in Figure 7.5(right).

Pixels of the virtual camera image that reproject outside the foreground silhouette of at least one of the input images are immediately rejected and all further operations are automatically discarded by the GPU hardware. This is the case for pixel $q_v$ in Figure 7.5(left). This provides a means to lever both speed and quality because noise in the segmentation silhouettes will, with high probability, not be available in all $N$ cameras.

If pixel $p_v$ is not discarded for plane depth $z_j$, its mean (i.e. interpolated) color $\psi_j$ and joint matching cost $\varepsilon_j$ are computed:

$$\psi_j(p_v) = \sum_{i=1}^{N} \frac{I_i(p_i)}{N} \tag{7.8}$$

$$\varepsilon_j(p_v) = \sum_{i=1}^{N} \frac{\left\| \psi_j(p_v) - I_i(p_i) \right\|^2}{3N} \tag{7.9}$$

where pixel $p_v$ reprojects to pixel $p_i$ according to Equation 7.5 and Equation 7.6.

To finally produce the interpolated image $I_v$ and joint depth map $Z_v$, the plane is swept for all $M$ depths $\{z_1, \ldots, z_M\}$ and the best match is selected on a winner-takes-all (WTA) basis:

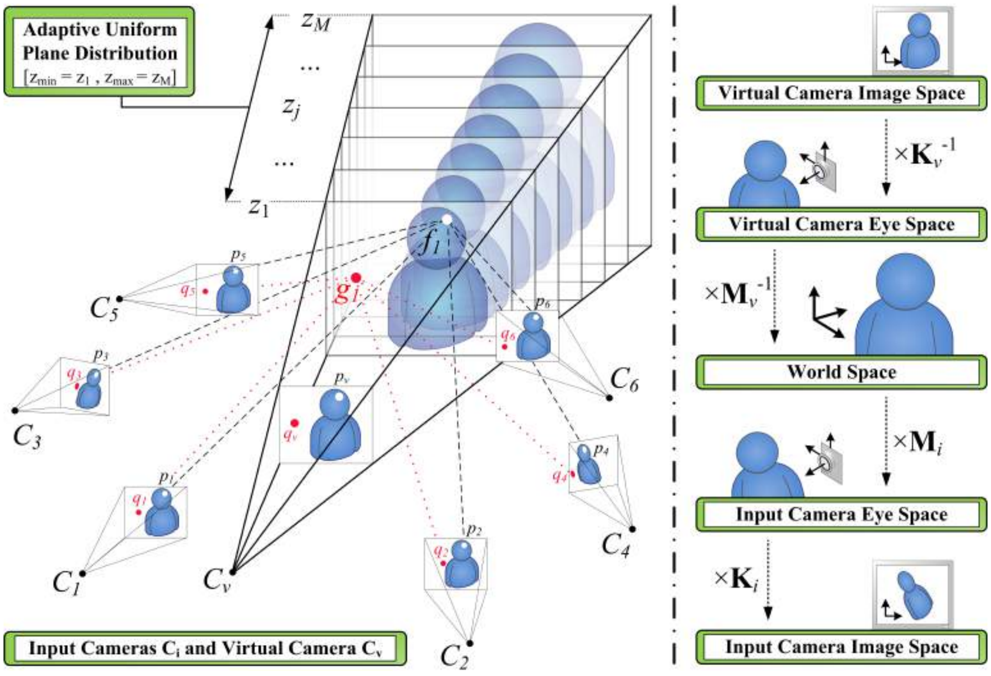


**Figure 7.5:** Concept of our plane sweeping algorithm that takes into account input image segmentation to discard unwanted computations. (left) Pixel $q_v$ is discarded because it deprojects to voxel $g_1$ on plane depth $z_1$ and subsequently projects to pixel $q_i$ outside of the foreground silhouette of at least one input image. Pixel $p_v$ does project inside all foreground silhouettes, via voxel $f_1$. (right) The relation between coordinate frames to project pixels from the virtual camera image to the input camera images.

$$I_v(p_v) = \underset{\psi_j(p_v)\ |\ z_j \in \{z_1,\dots,z_M\}}{\arg\min} \varepsilon_j(p_v) \tag{7.10}$$

$$Z_v(p_v) = \underset{z_j \in \{z_1,\dots,z_M\}}{\arg\min} \varepsilon_j(p_v) \tag{7.11}$$

The result is shown in Figure 7.6.

Similar to Equation 5.19 and Equation 5.20, p. 80, we can also keep a final confidence map $\varepsilon_v$:

$$\varepsilon_v(p_v) = \min_{z_j \in \{z_1,\dots,z_M\}} \varepsilon_j(p_v) \tag{7.12}$$

Contrary to Yang et al. [2002], in Equation 7.9 we propose to use all input cameras to compute the matching cost. By matching $N = 6$ cameras at once, the reliability of a pixel-to-pixel match is significantly increased and the need for very expensive cost aggregation

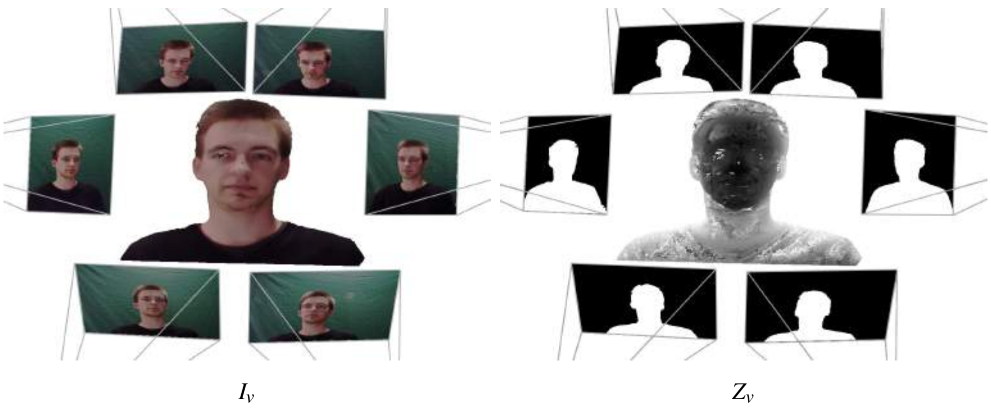


**Figure 7.6:** The view interpolation module outputs an interpolated image $I_v$ and its joint depth map $Z_v$. The interpolated image is shown surrounded by the input images $I_i$ and the depth map by the segmentation silhouettes $S_i$.

is eliminated. This is in stark contrast to the stereo matching of chapter 5, where the cost aggregation demanded over 80% of the total processing time (see Table 5.2, p. 90). The view interpolation is thus kept extremely efficient, which frees up resources for extensive post-processing in the upcoming depth refinement module. Should cost aggregation nevertheless be desired, an efficient way of doing so is to use OpenGL's mipmapping capabilities on each plane's confidence map $\varepsilon_j$ (Equation 7.9) to hierarchically aggregate costs over variable window sizes [Yang and Pollefeys, 2003; Yang et al., 2004b].

### 7.2.3 Depth Refinement

The interpolated image $I_v$ still contains photometric errors due to mismatches caused by variations in illumination, partially occluded areas and natural homogeneous texture of the human face. They are more easily detectable and correctable in the joint depth map $Z_v$, if we assume that it must be locally linear. We discern two specific types of errors to treat: erroneous patches (section 7.2.3.1) and speckle noise (section 7.2.3.2). The result is a refined depth map $\tilde{Z}_v$ which is, however, no longer linked to the interpolated image $I_v$. This link will be restored by the recoloring module in section 7.2.4. A step by step overview is given in Figure 7.8.

#### 7.2.3.1 Erroneous Patch Filtering

To detect erroneous patches, we propose the spatial filter kernel depicted in Figure 7.7(a). The entire following process is conceptually explained in Figure 7.7(b)–(e).

First, on the area around every pixel $p_v = (x_v, y_v)$ of the depth map $Z_v$, a two-dimensional depth consistency check is performed:

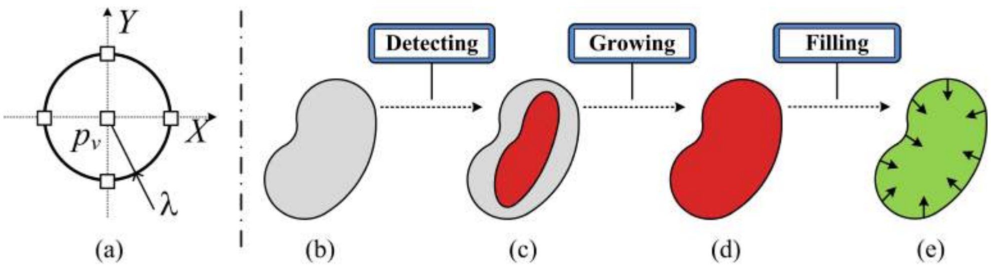


**Figure 7.7:** The depth refinement module uses a spatial filter kernel to detect and correct depth inconsistencies. (left) The proposed filter kernel and (right) the outlier correction concept, whereby (b)–(c) erroneous patches are detected (red), (c)–(d) grown outward, and (d)–(e) filled inward with reliable neighboring depth values (green).

$$||Z_v(x_v-\lambda,y_v)-Z_v(x_v+\lambda,y_v)||^2<\delta_c \tag{7.13}$$

$$||Z_v(x_v,y_v-\lambda)-Z_v(x_v,y_v+\lambda)||^2<\delta_c \tag{7.14}$$

where $\delta_c$ is a small threshold to represent the depth consistency and $\lambda$ determines the maximum size of patches that can be detected.

If the area passes the consistency check in either one of the dimensions, the pixel $p_v$ is flagged as an outlier if $p_v$ itself does not exhibit the same consistency by exceeding a given threshold $\delta_o$:

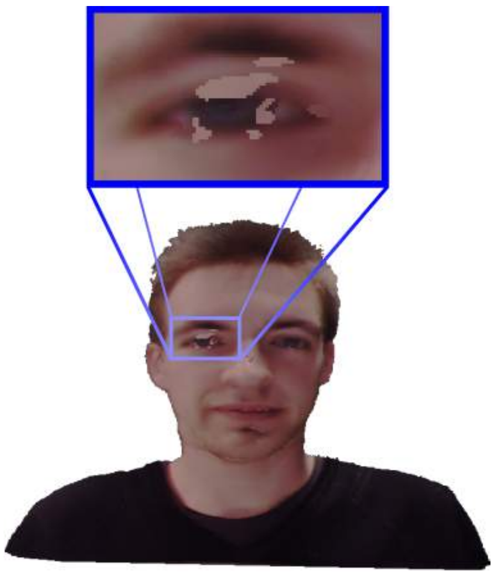
$$\left\|Z_v(x_v,y_v)-\frac{Z_v(x_v-\lambda,y_v)+Z_v(x_v+\lambda,y_v)}{2}\right\|^2>\delta_o \tag{7.15}$$

$$\left\|Z_v(x_v,y_v)-\frac{Z_v(x_v,y_v-\lambda)+Z_v(x_v,y_v+\lambda)}{2}\right\|^2>\delta_o \tag{7.16}$$

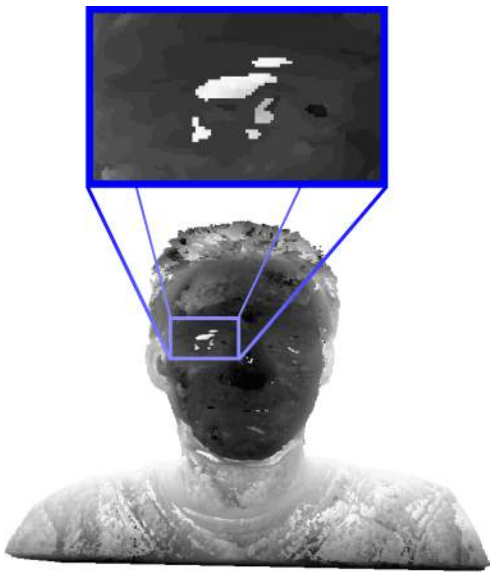
Next, the detected center is grown only if its neighboring pixels exhibit the same depth consistency as the initial outliers and the full patch is thereby detected.

Finally, the direction of growth is reversed and the detected patch is filled with reliable depth values from its neighborhood. Detected erroneous patches and the corrected depth map are shown in Figure 7.8(middle row).

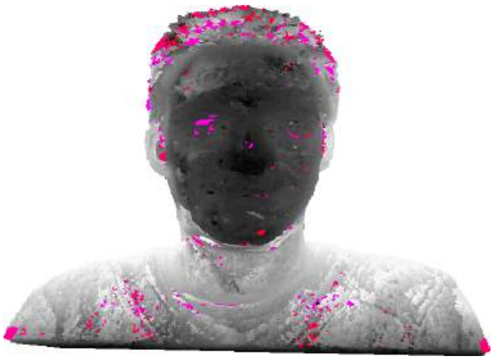
**Figure 7.8 *(facing page)*:** The initial interpolated image $I_v$ still contains erroneous patches and speckle noise. The depth refinement module takes the initial depth map $Z_v$ as input. First, erroneous patches are detected and corrected. Next, high-frequency speckle-noise is removed by applying a Gaussian low-pass filter. The result is the refined depth map $\tilde{Z}_v$, which will ultimately be recolored by the recoloring module to produce the final interpolated image $\tilde{I}_v$.



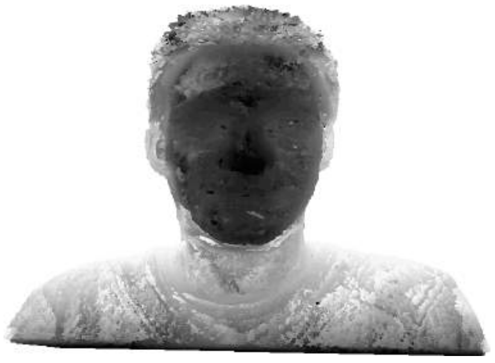
$I_v$



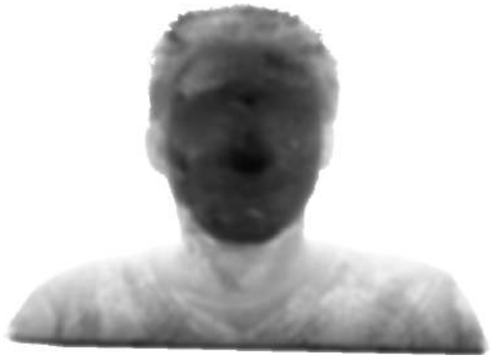
$Z_v$



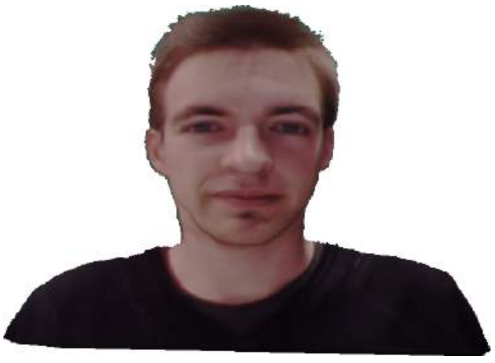
$Z_v$: erroneous patches detected



$\tilde{Z}_v$: erroneous patches corrected



$\tilde{Z}_v$: speckle noise filtered



$\tilde{I}_v$

Since all of these procedures operate iteratively and pixel-wise, they are inherently appropriate for parallel implementation on a GPU, thereby again achieving a tremendous speedup over a generic CPU algorithm.

#### 7.2.3.2 Speckle Noise Filtering

The human face naturally contains large homogeneously textured regions with skin color. These areas are easily mismatched, which causes the depth map to contain spatial high frequency speckle noise. This type of noise is most efficiently suppressed by a low-pass filter and we therefore apply a standard two-dimensional isotropic Gaussian filter. The final refined depth map $\tilde{Z}_v$ is shown in Figure 7.8(bottom-left).

Although applying a low-pass filter eliminates the geometrical correctness of the depth map, this still significantly enhances the subjective visual quality, as opposed to other more geometrically correct approaches [Lei and Hendriks, 2002]. Furthermore, a Gaussian filter can be orthogonally separated into a one-dimensional row and column convolution, which again allows for a highly optimized implementation on graphics hardware.

To conclude, note that the refined depth values are not necessarily elements of the initial collection of uniformly distributed depth values $\{z_1, \ldots, z_M\}$anymore. Care should be taken to ensure that they are still at least within the range $[z_{min}, z_{max}]$, as this will be a requisite for the upcoming complexity control module in section 7.2.5.

### 7.2.4 Recoloring

The depth refinement changes the geometry of the depth map $Z_v$, which is normally – inherent to the plane sweep – jointly linked to the interpolated image $I_v$. To restore this link, we will require the ability to reproject the pixels (depth values) of the refined depth map $\tilde{Z}_v$ to pixels in the input images $I_i$:

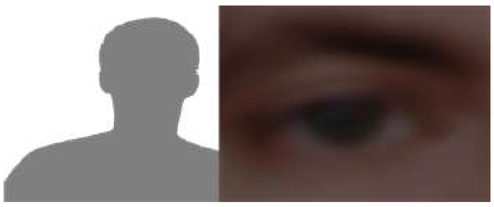
$$f_v = M_v^{-1} \times K_v^{-1} \times \tilde{T}_v \times p_v \tag{7.17}$$

$$p_i = K_i \times M_i \times f_v \tag{7.18}$$

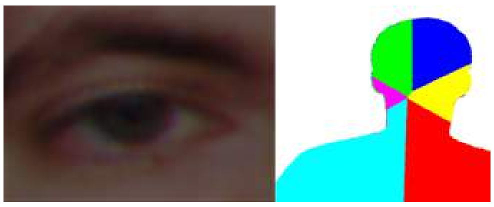
which is the exact same process as Equation 7.5 and Equation 7.6, but with a different translation and scaling matrix $\tilde{T}_v$ that incorporates the reprojected pixel's depth value $\tilde{Z}_v(p_v)$:

$$\tilde{T}_v = \begin{bmatrix} \tilde{Z}_v(p_v) \times X_{res} & & & 0 \\ & \tilde{Z}_v(p_v) \times Y_{res} & & 0 \\ & & 1 & \tilde{Z}_v(p_v) \\ & & & 1 \end{bmatrix} \tag{7.19}$$

so that the world space position of voxel $f_v$ for pixel coordinate $p_v$ is now determined by its refined depth value $\tilde{Z}_v(p_v)$, instead of by a given plane depth $z_j$ as in Equation 7.7.



(a) N-camera recoloring.



(b) Confident-camera recoloring.

**Figure 7.9:** The recoloring module recolors the refined depth map to produce the final interpolated image. (a) N-camera recoloring equally blends all $N$ input cameras. (b) Confident-camera recoloring selects a single input camera that is closest in angle to the virtual camera, determined using the color-coded index map in which each color represents another input camera.

We next devise two methods to resolve the color of the final interpolated image $\tilde{I}_v$: N-camera recoloring (section 7.2.4.1) and confident-camera recoloring (section 7.2.4.2). Each method has its particular effect on the visual quality.

### 7.2.4.1 N-Camera Recoloring

This simplest and fastest recoloring method obtains the interpolated pixel color by blending all $N$ cameras:

$$\tilde{I}_v(p_v) = \sum_{i=1}^{N} \frac{I_i(p_i)}{N} \tag{7.20}$$

where pixel $p_v$ of the interpolated image $\tilde{I}_v$ reprojects to pixel $p_i$ in the input image $I_i$ according to Equation 7.17 and Equation 7.18.

As Figure 7.9(a) shows, this approach generates very smooth transitions of the input images in the recolored result, at the expense of loss of detail.

### 7.2.4.2 Confident-Camera Recoloring

For each pixel $p_v$ of the image $\tilde{I}_v$, the second recoloring method determines which input camera $C_i$ is closest in angle to the virtual camera $C_v$ and stores that information in an index map $K$:

$$K(p_v) = \underset{i \in \{1,\ldots,N\}}{\arg\max} \cos\left(\widehat{\vec{k}_v \vec{k}_i}\right) \tag{7.21}$$

where we search for the largest cosine of the angle between the vector $\vec{k}_v = f_v - C_v$ from the virtual camera center $C_v$ to the voxel $f_v$ and the vector $\vec{k}_i = f_v - C_i$ from the input camera center $C_i$ to the same voxel $f_v$. We assume $C$ to represent the optical center of the camera in world space, while the voxel $f_v$ is the pixel coordinate $p_v$ deprojected to world space according to Equation 7.17.

Selecting the final color from a single camera designated by the index map $K$ ensures a sharply detailed recolored image:

$$\tilde{I}_v(p_v) = I_k(p_k) \tag{7.22}$$

with index $k = K(p_v)$. However, the quality is sensitive to inaccuracies in the refined depth map and to deviating colors between input cameras due to variations in illumination and photometric calibration (section 2.4.2, p. 28). This recoloring scheme is illustrated in Figure 7.9(b), together with its color-coded index map.

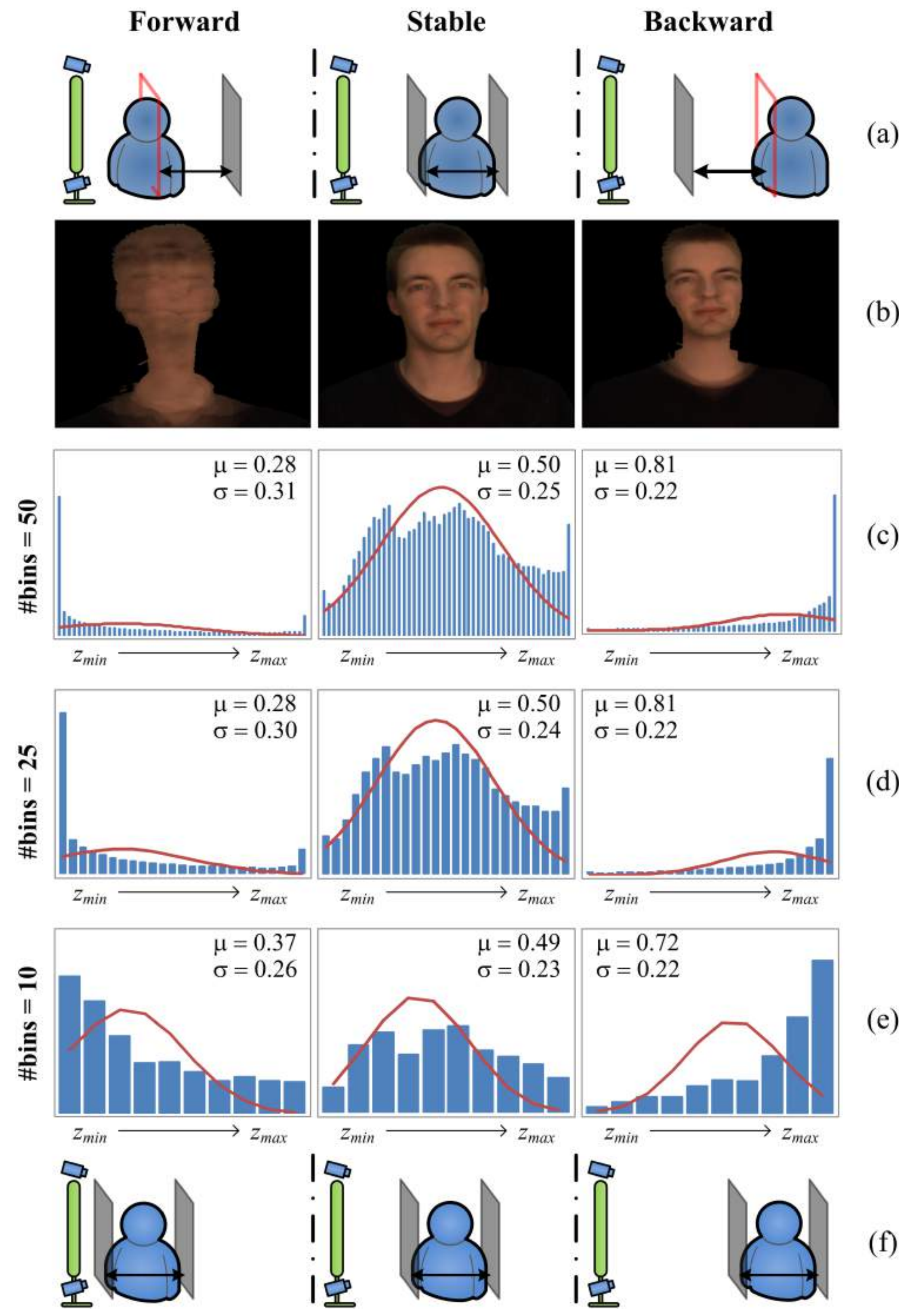
This approach can also be interpreted as a simplified version of unstructured lumigraph rendering [Buehler et al., 2001], where the refined depth map $\tilde{Z}_v$ would serve as a geometry proxy and where the index map $K$ would incorporate angular, resolution, field of view and visibility constraints.

### 7.2.5 Complexity Control: Dynamic Uniform Plane Distribution

To avoid heavy constraints on the user's freedom of movement, a large depth range needs to be swept. This implies a lot of redundant computations, since in actuality the head of the user only occupies a small range. On the other hand, simply keeping the depth range fixed to a narrow range will not do either. The user could then easily move out of range and this would have a distorting effect on the interpolated image, as shown in Figure 7.10 rows (a)–(b). Instead, we would like a narrow depth range to stay focused around the user's head while he moves, as depicted in Figure 7.10 row (f).

At the end of each iteration of the processing chain, the new range for the next iteration can be determined. This is done by analyzing the histogram $H(z)$ on the depth values of the refined depth map $\tilde{Z}_v$, which was computed over the current range $[z_{min}, z_{max}]$. Three separate cases can be distinguished, shown in Figure 7.10 row (c), as the user moves in front of the screen:

**Figure 7.10** ***(facing page)*****:** Our histogram analysis scheme for dynamic uniform plane depth distribution in plane sweeping. (a) The user can easily move out of a fixed narrow range, (b) which has a distorting effect on the interpolated image. (c) If the user moves forward the histogram of the depth map will contain an exceptionally large number of depth values toward $z_{min}$. Likewise for $z_{max}$ if the user moves backward. The depth range for the next iteration can be determined from a Gaussian fit $G(\mu, \sigma)$ on the histogram, by Equation 7.23. (d)–(e) Approximations to $G(\mu, \sigma)$ can be made by reducing the number of histogram bins and the number of depth values to bin. (f) The result is a narrow depth range that stays focused around the user.


Forward
Stable
Backward
(a)
(b)
(c)
(d)
(e)
(f)
#bins = 50
#bins = 25
#bins = 10
μ = 0.28
σ = 0.31
μ = 0.50
σ = 0.25
μ = 0.81
σ = 0.22
μ = 0.28
σ = 0.30
μ = 0.50
σ = 0.24
μ = 0.81
σ = 0.22
μ = 0.37
σ = 0.26
μ = 0.49
σ = 0.23
μ = 0.72
σ = 0.22
$z_{min} \longrightarrow z_{max}$

- *Forward Movement*: If the user moves forward, he will exit the depth range that is currently being swept. Therefore, the histogram will contain an exceptionally large number of depth values toward $z_{min}$.
- *No Movement (Stable)*: When the user keeps his head more or less motionless, the histogram will contain a clear peak in the middle.
- *Backward Movement*: Analogous to forward movement, the histogram will show a peak toward $z_{max}$ if the user moves out of range backward.

For practical applicability and generality of parameters, the histogram is in fact computed on the depth values after they have been normalized to the range $[0,1]$. We next fit a Gaussian distribution $G(\mu,\sigma)$ with center $\mu \in [0,1]$ and standard deviation $\sigma \in [0,1]$ on the normalized histogram. These parameters are also printed in Figure 7.10 row (c). The current range $[z_{min}, z_{max}]$ is then updated according to:

$$\begin{aligned} z'_{min} &= z_{min} + (\mu - b_1\sigma)(z_{max} - z_{min}) \\ z'_{max} &= z_{min} + (\mu + b_2\sigma)(z_{max} - z_{min}) \end{aligned} \tag{7.23}$$

where $[z'_{min}, z'_{max}]$ is the resulting depth range for the next iteration, and $b_1$ and $b_2$ are constant forward and backward bias factors that can be adapted to the inherent geometry of the object in the scene.

As the user moves forward or backward, the center $\mu$ of the Gaussian fit changes and dynamically adapts the effective depth range of the system. For the next iteration, all $M$ plane depths are uniformly distributed in the range $[z_1 = z'_{min}, z_M = z'_{max}]$. In other words, the circle is completed by inputting $z'_{min}$ and $z'_{max}$ in Equation 7.4.

As expressed by the inversely proportional relation in Equation 2.1, p. 15, choosing a uniform distribution of plane depths results in a non-uniform distribution of disparities (and vice versa) [Chai et al., 2000; Feldmann et al., 2003], which in turn risks undersampling the image resolution. Our approach presented here, however, not only leverages the dynamic range of the plane distribution in space, but also implicitly increases the accuracy by significantly reducing the chance at mismatches.

The histogram – and with it the parameters of the Gaussian fit – can be approximated by reducing the number of histogram bins and the number of depth values to bin. This is illustrated in Figure 7.10 rows (d)–(e), but we will examine this closer when we discuss implementation-specific optimizations in section 7.3.

Our approach proposed here is similar to the one we employed earlier to increase the performance and quality of our stereo matching algorithm. In section 6.2, p. 105, we analyzed the disparity map histogram $H(d)$ to restrict the disparity search range. However, the matching algorithm had to be executed twice per frame: once at reduced resolution and once at full resolution. Our approach here does not introduce extra iterations of the view interpolation.

It does, however, come at the expense of being behind one frame in responsiveness, as the depth range for the next iteration is determined at the end of the current iteration. Exactly because this update is always one frame behind, the interpolated image is briefly distorted if the user moves out of range, but will quickly recover as the range is adapted. A real-time high frame rate therefore increases the responsiveness of the system and allows it to stabilize quickly. Consequently, movement at normal moderate speed will not be visually noticed by the users.

### 7.2.6 Eye Tracking For Virtual Camera Positioning

Eye contact will only be convincing if the virtual camera at the local site accurately represents the eyes of the remote user, and vice versa, as illustrated in Figure 7.11(a). However, for eye gaze to be correct, the position and orientation of the virtual camera must be expressed in 3D and cannot be defined directly on the screen in the eyes of the remote user [Waizenegger et al., 2012]. The eye tracking method that we present here tracks the user's eyes in 3D and determines their position relative to the screen. It is suitable for robust and efficient implementation on the CPU and can thus run concurrently with the main view interpolation on the GPU. The algorithm is described in detail by Maesen [2016].

First we detect skin and eye candidates in every input image. Each input image is transformed to the YCbCr color space, with Cb and Cr the blue-difference and red-difference chroma components respectively. The luminance (brightness) component Y is redundant for our purposes, as brightness does not contribute to characterizing skin colors under normal lighting conditions. A pixel is marked as a skin pixel if it complies to:

$$\begin{gathered} |Cb_i - \sigma_b| < \tau_s \\ \text{and} \\ |Cr_i - \sigma_r| < \tau_s \end{gathered} \tag{7.24}$$

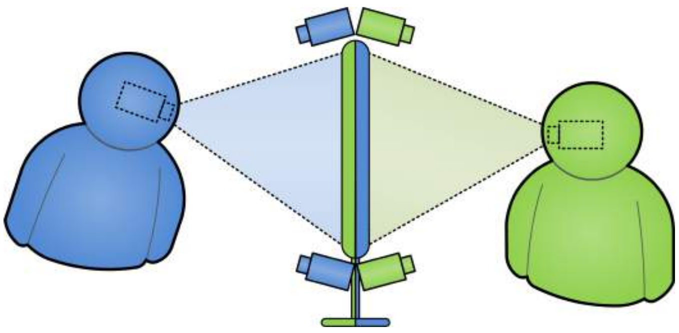
with $Cb_i$ and $Cr_i$ the chroma components of the input image $I_i$ and normalized to the range $[0,1]$, and with $\tau_s$ a given threshold. The parameters $\sigma_b$ and $\sigma_r$ identify the subspace associated with skin color. It has been shown that in the CbCr subspace the human skin color forms a compact class that can be modeled by a 2D Gaussian distribution [Bergasa et al., 2000; Hsu et al., 2002]. For our purposes, however, the 1D approximation above suffices.

Eye candidates are detected in the same color space. A pixel is marked as an eye candidate if it complies to [Hsu et al., 2002]:

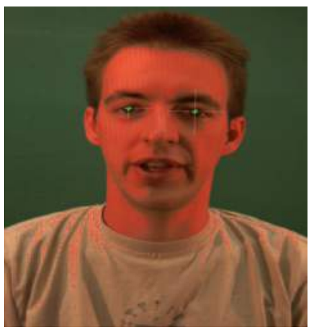
$$\frac{1}{3}\left((Cb_i)^2 + (1 - Cr_i)^2 + (Cb_i/Cr_i)\right) > \tau_e \tag{7.25}$$

with $\tau_e$ again a given threshold.

Lastly, the eye candidates are clustered and the cluster means are validated by cross-checking them over their epipolar geometry between all input images and by requiring them



(a) The local user moves and the remote virtual camera must follow.



(b) Skin and eye pixels.

**Figure 7.11:** (a) Whenever the local user moves, the virtual camera at the remote side – a substitute for the local user's real eyes at the remote side – needs to move accordingly to ensure that the virtual image is reconstructed from the correct viewpoint. Conceptually, the two screens are pasted back-to-back against each other, with both users peering into each other's world. (b) The virtual camera is positioned by tracking the user's eyes. As a first step, skin (red) and eye (green) pixels are detected in each input image. The procedure is described in detail by Maesen [2016].

to be surrounded by skin pixels. Detected skin and eye pixels in one input image are marked in Figure 7.11(b).

Once the eye coordinates of the local user have been located in every input image, they are triangulated to 3D coordinates in world space. Next, the triangulated coordinates are expressed in a local coordinate frame relative to the screen. They are then sent over the network to the remote side, where they are mirrored toward the screen to determine the position of the virtual camera. An orientation is added by requiring the virtual camera to point (i.e. look) at the middle of the screen, while its field of view is bounded by the screen's corners. The two screens (one at each user's side) are placed in a common (metric) coordinate frame, as if they were pasted back-to-back against each other. This creates the immersive effect of a virtual window into the world of the remote user.

### 7.2.7 Networking

Our prototype system sends the local user's eye coordinates over the network, allowing the eye gaze corrected image to be synthesized locally at the remote user's side where the relevant images are captured. The total peer-to-peer communication thus consists only of the interpolated images and the eye coordinates. This avoids the transfer of all $N$ input images and brings the required network communication to a minimum. Hence, data processing and communication are optimized to guarantee the real-time aspect of the system, even when running on inexpensive commodity hardware.

## 7.3 Implementation and Optimizations

The modular design of our prototype allows us to carefully select independent algorithms that each efficiently exploit the GPU for general-purpose computations. In this section, we take a closer look at some of the design and implementation choices that we made to harness the powerful computational resources of the graphics hardware and the optimizations that follow from them.

First, the arithmetic intensity of the algorithm is maximized to ensure real-time performance in section 7.3.1. Next, the system is further accelerated by elevating the processing granularity from pixels to tiles in section 7.3.2. Lastly, our just-in-time approach to GPU code compilation is explained in section 7.3.3.

### 7.3.1 Increased Arithmetic Intensity

Slow download and readback speed to and from the GPU is a frequent bottleneck in many real-time applications. The culprit is the memory bandwidth provided by PCIexpress, the bus connection between the motherboard north bridge controller and the GPU. We can tackle this bottleneck by directly retrieving one-channel Bayer patterns from the cameras and transferring these instead to the GPU memory, which reduces the bandwidth consumption by a factor three. For the simultaneous download of $N = 6$ input images the impact on the overall system performance is not to be underestimated, as is clearly understood by comparing the left and right pie charts in Figure 7.12. The download time is reduced from 12.78 ms to 3.08 ms, which allows the frame rate to increase with no less than 35% from 34 FPS to 46 FPS.

Of course, as a consequence, the preprocessing module now requires an additional Bayer demosaicing step. We explained Bayer demosaicing in section 2.4.4, p. 30, with an example shown in Figure 2.12, p. 30. By default we use the bilinear demosaicing method [Ramanath et al., 2002], because it lends itself very well for efficient pixel-wise implementation on the GPU. More complex methods are available also [Malvar et al., 2004; Hirakawa and Parks, 2005; Goorts et al., 2012b].

Naturally, more online computations are introduced by inserting an additional demosaicing processing step. In this case, however, this increased arithmetic intensity is desirable and not detrimental to the overall performance. The reason is that graphics hardware benefits greatly from kernels with high arithmetic intensity, as it processes instructions significantly faster than it transfers data.

Processing time measurements of all other modules are also available in Figure 7.12. We will have a detailed look at these when we discuss results in section 7.4.

### 7.3.2 Elevated Granularity

Commonly, the programmable graphics rendering pipeline, as depicted in Figure 2.7, p. 22, can be exploited for pixel-wise general-purpose computations as follows. A single quad

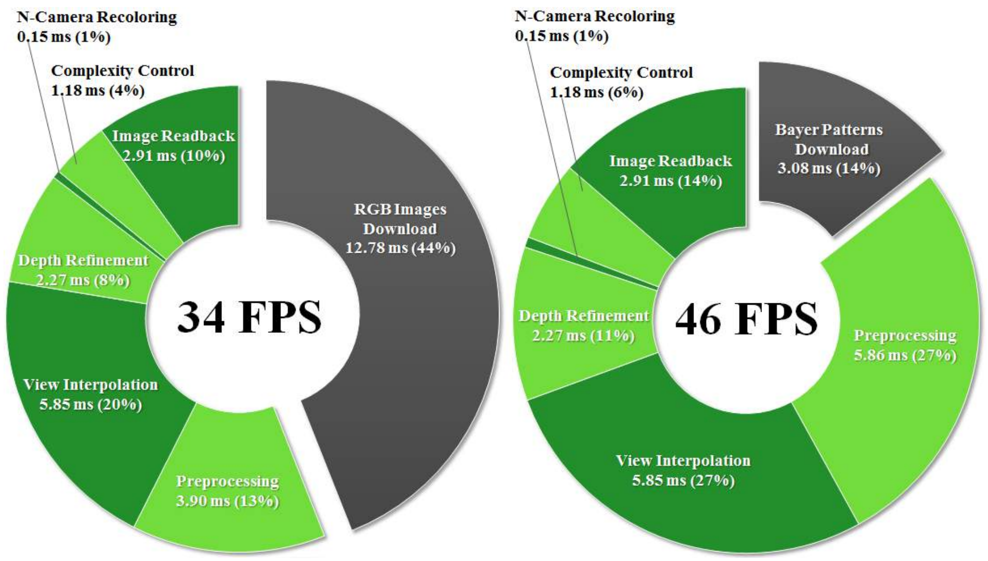


**Figure 7.12:** Workload profiling for (left) slow RGB images versus (right) fast Bayer patterns download. Downloading one-channel Bayer patterns instead of three-channel RGB images to the GPU means a significant reduction in bandwidth consumption and thus improves performance considerably. A more detailed workload profiling with Bayer patterns download is shown in Figure 7.17.

is drawn and the pipeline (matrix stack, viewport, etc.) is configured in such a way that the quad's four corner vertices project to the corners of an off-screen buffer with the same resolution as the image that we wish to process. Texturing the quad with the image then results in a pixel-to-pixel mapping between each texel (i.e. a pixel of the texture image) and each fragment (i.e. a pixel of the off-screen buffer). The hardware invokes a user-configurable program (the fragment shader) for each fragment, allowing us to perform our own pixel-wise computations on the image and write the result to the off-screen buffer. Suppose the fragment shader implements the green screening test of Equation 7.1, then each pixel can be tested in parallel as illustrated in Figure 7.13.

As there can be millions of fragments and the fragment shader runs independently for each and every one of them, the fragment processing stage is also the most performance-sensitive part of the graphics pipeline. Much of its functionality, however, can often be carried over to the preceding vertex processing stage. Again the hardware invokes a user-configurable program (the vertex shader) for each incoming vertex, in our case the quad's four corner vertices. By triangulating the quad into smaller tiles, we effectively elevate the processing granularity from pixels to tiles. Computations are only performed on the vertices (corner points) of the tiles and the result of pixels inside the tiles is approximated by hardware-inherent linear interpolation.

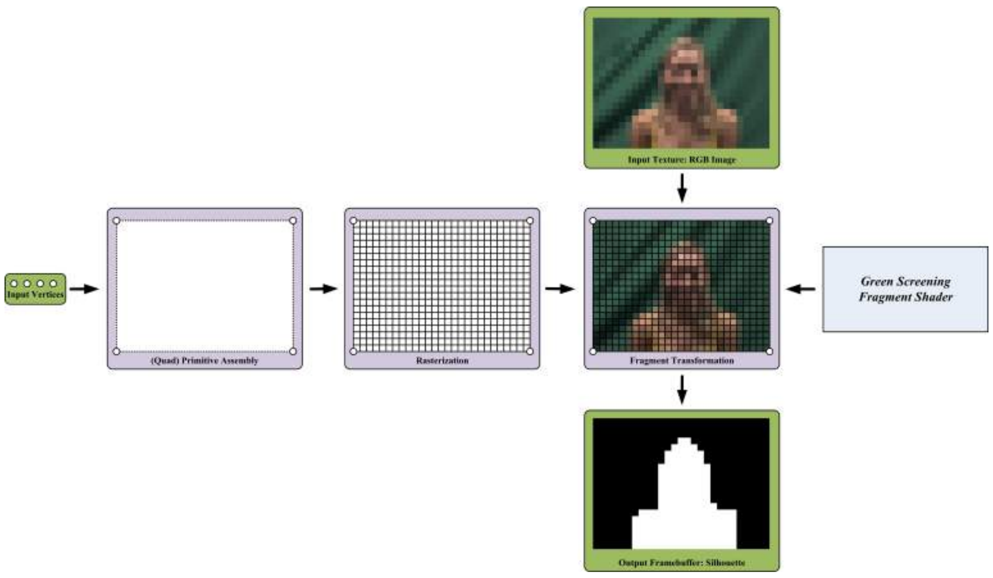

Input Texture: RGB Image
Input Vertices
(Quad) Primitive Assembly
Rasterization
Fragment Transformation
Green Screening Fragment Shader
Output Framebuffer: Silhouette

**Figure 7.13:** We can configure the graphics rendering pipeline of Figure 2.7, p. 22, in such a way that a fragment-to-texel mapping emerges between the fragments of a single rendered quad and the texels of an input texture. By implementing our own fragment shader functionality, massively parallel pixel-wise general-purpose computations on the input texture become possible. This example shows how to compute a segmentation silhouette of a $27 \times 20$ image by implementing Equation 7.1.

As a result, the computational complexity becomes inversely proportional to the granularity of the tessellation. If the tile size is chosen wisely, the overall execution speed can be drastically increased without noticeable impact on the visual quality. We implement three optimization schemes based on this speed versus quality trade-off. Their granularity can be configured by a set of well-defined parameters $0 < t \leq 1$.

**Tiled Undistortion** Lens distortion is generally corrected on a per-pixel basis by the preprocessing module of section 7.2.1. However, it can be efficiently approximated by triangulating (tiling) a quad, offsetting the vertices of the triangles according to the distortion model, and finally texturing the (un-)distorted quad with the image that is to be corrected. The subdivision resolution is configurable by a scaling factor $0 < t_u \leq 1$.

The lens correction is hence completely ported from the fragment to the vertex processing stage. The fragment stage becomes available to perform the subsequent image segmentation in a single pipeline pass, thereby significantly leveraging the GPU utilization.

**Tiled Tallying in Reduced Bins** For the complexity control module of section 7.2.5, a first approximation to the histogram can be made by reducing the resolution of the depth map to be binned. As such, the multi-resolution capabilities of the GPU are used to tally tiles instead of pixels in the histogram bins. This is expressed by a factor $0 < t_s \leq 1$ proportional to the sampling resolution.

Evidently, it is of no use to have more histogram bins than the number of planes (depth values $z_j$) in the sweep. The essential part, however, is deriving the parameters $\mu$ and $\sigma$ of the Gaussian fit to dynamically adjust the depth range by Equation 7.23. As illustrated in Figure 7.10 row (d), the histogram can be further approximated by reducing the number of bins, without a large impact on the Gaussian parameters. More specifically, the number of bins are determined proportionally to the number of planes $M$ by a factor $0 < t_b \leq 1$. Heavily reducing the number of bins in Figure 7.10 row (e) causes the center $\mu$ to become less accurate, as it is pulled toward the center of the depth range. An optimal trade-off point exists, since the accuracy loss will cause the responsiveness of the system to decrease.

Many options are available for efficient histogram binning on the GPU. We implemented and compared three, but found no significant difference in performance between the OpenGL GL_HISTOGRAM extension [Segal and Akeley, 1999], vertex scattering [Green, 2005] and occlusion querying [NVIDIA Corporation, 2005].

**Tiled Recoloring** For the confident-camera recoloring of section 7.2.4.2, the depth map can again be tessellated by a factor $0 < t_k \leq 1$. This significantly accelerates the construction of the index map $K$ (Equation 7.21) by interpolating angles between the tile corners.

### 7.3.3 Just-In-Time Compilation

We dynamically generate and compile the vertex and fragment shader code at run time, whenever a change is required in the code. This increases performance in many ways. First and foremost, loops can be unrolled and unnecessary branching is simply cut away. Furthermore, many parameters that otherwise would need to be passed by value can be compiled into the code. Just-in-time compilation is feasible because the system's parameters rarely change once they are configured.

## 7.4 Results

We built our prototype by mounting $N = 6$ auto-synchronized Point Grey Grasshopper cameras on a custom-made frame that closely surrounds the screen, as shown in Figure 7.2. This camera setup avoids large occlusions and has the potential to generate high quality eye gaze corrected images, since no image extrapolation is necessary. We calibrate the cameras offline as described in section 2.4.1, p. 26, and if necessary maintain the external calibration using the procedure developed in chapter 3. However, in an end product the cameras could be built into the monitor bezel, which would fix the camera parameters and make extensive calibration superfluous.

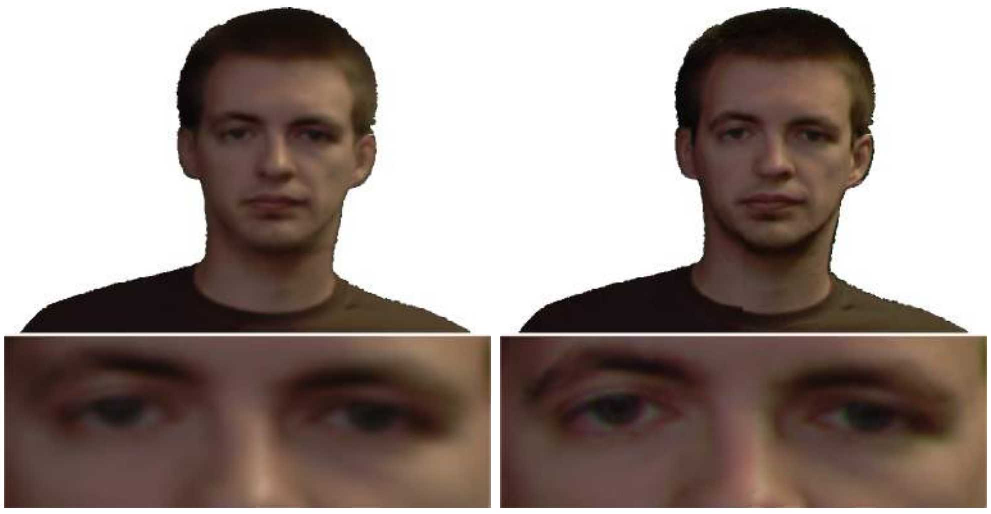

(a) N-camera recoloring. (b) Confident-camera recoloring.

**Figure 7.14:** Eye gaze corrected images using (a) N-camera versus (b) confident-camera recoloring.

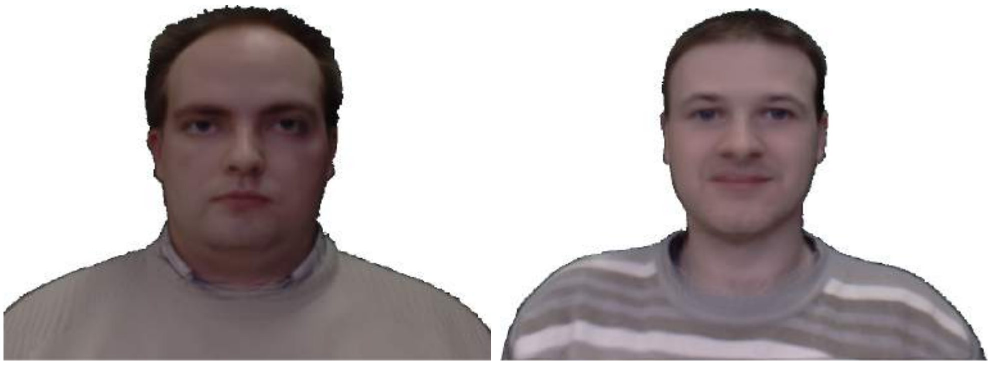
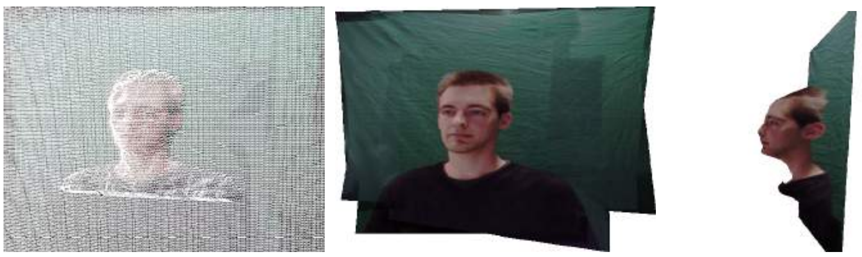

Steven Sammy

**Figure 7.15:** More eye gaze corrected images using N-camera recoloring.

A final eye gaze corrected result that compares both recoloring schemes is shown in Figure 7.14. For the N-camera recoloring, some small artifacts along the ears and chin, together with minor ghosting around the neck, remain noticeable due to limitations of the depth refinement. The result generated with the confident-camera recoloring is sharper and much more detailed, although some minor artifacts are still caused by abrupt camera transitions in the index map $K$ (Equation 7.21). More results for different users are shown in Figure 7.15. All images maintain their integrity, are regarded to be of high subjective visual quality and convincingly seem to be making eye contact with the reader.



(a) Triangulated proxy. (b) Moderately sideways. (c) Extremely sideways.

**Figure 7.16:** Unstructured lumigraph rendering from non eye gaze corrected viewpoints. The refined depth map has been triangulated into (a) a geometry proxy, which has then been rendered from (b) a moderately deviating viewpoint and (c) an extremely deviating viewpoint. In particular the latter shows that the refined depth map approximates the true shape of the face well.

Figure 7.16 shows an unstructured lumigraph rendering [Buehler et al., 2001] (see section 2.1.3, p. 18) of a user in front of our prototype. First, we use our plane sweeping algorithm to compute the refined depth map $\tilde{Z}_v$ of the eye gaze corrected viewpoint. Next, from the depth map in image space, we triangulate a geometry proxy in world space. Finally, we render the proxy from an alternate (not eye gaze corrected) viewpoint and apply the unstructured lumigraph algorithm to take into account angular, resolution, field of view and visibility penalties. The figure allows us to judge that the geometric correctness of the refined depth map is very acceptable.

All these results are generated under moderately varying ambient conditions (e.g. changing illumination), but with the fixed set of fine-tuned parameters grouped in Table 7.1. We observed this configuration of parameters to behave well for various types of users, thus further demonstrating the genuine practicality of our prototype.

Processing time measurements were already previewed in Figure 7.12 (belonging to section 7.3.1), where we compared the impact of downloading three-channel RGB images with one-channel Bayer patterns on the system performance. A more detailed workload profiling is shown in Figure 7.17, with the N-camera recoloring used in Figure 7.12(right) replaced with the sharper, but more expensive, confident-camera recoloring.

We make some observations on the processing times of the different modules. Reading back a single eye gaze corrected RGB image takes more or less the same time as downloading $N = 6$ one-channel Bayer patterns, i.e. about 13% of the total processing time. The preprocessing and view interpolation modules are the most expensive, as together they consume half of the total processing time (25% each). The depth refinement and recoloring module only require about 10% each. The confident-camera recoloring is, however, remarkably more expensive than the N-camera recoloring: 2.52 ms (11%) versus only 0.15 ms (1%). In the end, the complexity control module turns out to be the least expensive, as it takes just 5%

| Module | Parameter | Value |
|---|---|---|
| Preprocessing | $\rho_g$ | 0.355 |
| | $\tau_f$ | 0.010 |
| | $\tau_b$ | 0.002 |
| | $\tau_a$ | 0.998 |
| | $t_u$ | 0.2 |
| View Interpolation | $N$ | 6 |
| | $M$ | 35 |
| Depth Refinement | $\lambda$ | 20 |
| | $\delta_c$ | 0.2 |
| | $\delta_o$ | 0.3 |
| Confident-Camera Recoloring | $t_k$ | 0.2 |
| Complexity Control | $b_1$ | 2.0 |
| | $b_2$ | 2.0 |
| | $t_s$ | 0.5 |
| | $t_b$ | 0.4 |
| Eye Tracking | $\sigma_b$ | 0.352 |
| | $\sigma_r$ | 0.473 |
| | $\tau_s$ | 0.017 |
| | $\tau_e$ | 0.446 |

**Table 7.1:** Fixed set of parameters for genuine practical usage of our prototype for close-up one-to-one eye gaze corrected video conferencing. We observed these parameters to behave well for various types of users under varying ambient conditions. This demonstrates the real-life practicality of our prototype.

of the total processing time. This is a great testament to its strength, as it itself prevents the already time-consuming view interpolation module from gobbling up even more time.

Processing time was measured on an NVIDIA GeForce 8800 GTX graphics card, with SVGA ($800 \times 600$) input/output resolution and parameters configured as in Table 7.1. Summing up the individual modules, we reach a confident speed of 42 FPS. However, our experimental setup is limited by 30 Hz support in the cameras and FireWire controller hardware. Still, we foresee these specifications to enable genuine practical usage, since the end-to-end system is developed with minimal constraints. Compared with stereo interpolation in chapter 6, our solution presented here is much more performant. Moreover, the system's current speed offers room for further quality improvements by advancing the algorithm and computational complexity.

Our system supports many-to-many communication, although some limitations apply. First, a larger screen is needed to show multiple users side by side. Consequently, the cameras are spaced further apart, which in turn requires a larger user-to-screen distance to alleviate occlusions. Furthermore, the complexity control module as it was developed in section 7.2.5

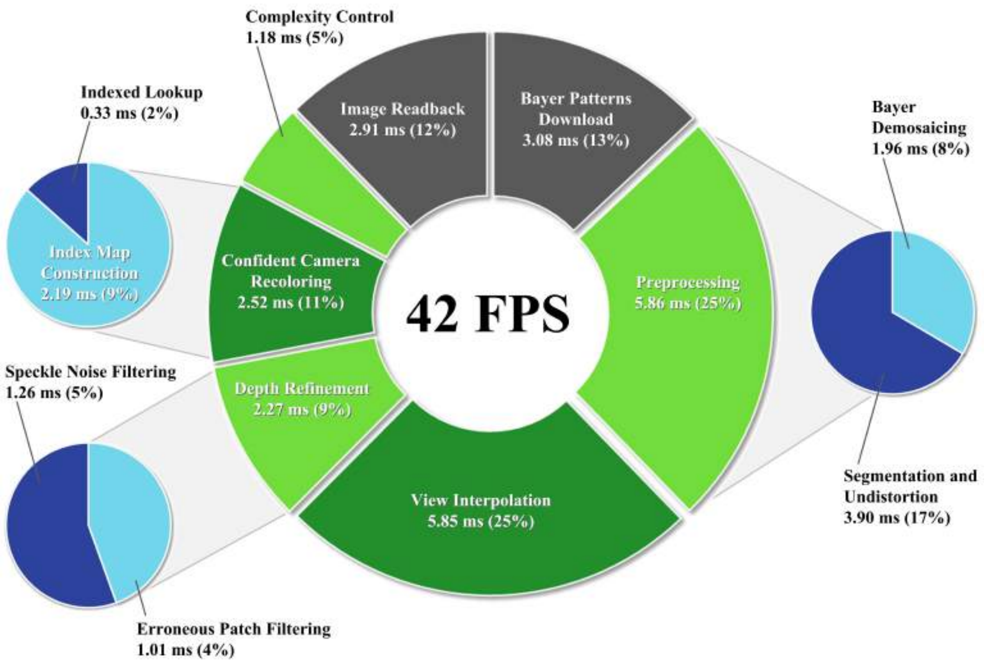


**Figure 7.17:** Detailed workload profiling of the end-to-end optimized processing chain. Processing time was measured on an NVIDIA GeForce 8800 GTX graphics card (Tesla architecture) [Lindholm et al., 2008], with an SVGA ($800 \times 600$) input/output resolution and parameters configured as in Table 7.1. Compared with Figure 7.12(right), we have opted here to use the more expensive confident-camera recoloring instead of N-camera recoloring.

only supports depth range adaptation for one object (i.e. one user) in the scene. This forces the users to remain more or less stationary and side by side at the same depth in the scene. In section 8.3, p. 170, we will extend the complexity control module to support multiple objects in the scene at different depths. Lastly, the synthesized image can naturally provide eye contact with only one user at the same time. Four users are experiencing our prototype simultaneously in Figure 7.18.

## 7.5 Requirements Evaluation

As is familiar by now, we here discuss to what degree our solution complies to the requirements specified in section 1.1, p. 3. The scores correspond to the scale defined there and are accompanied by the scores of environment remapping (chapter 4) and stereo interpolation (chapter 6) in Figure 7.19.

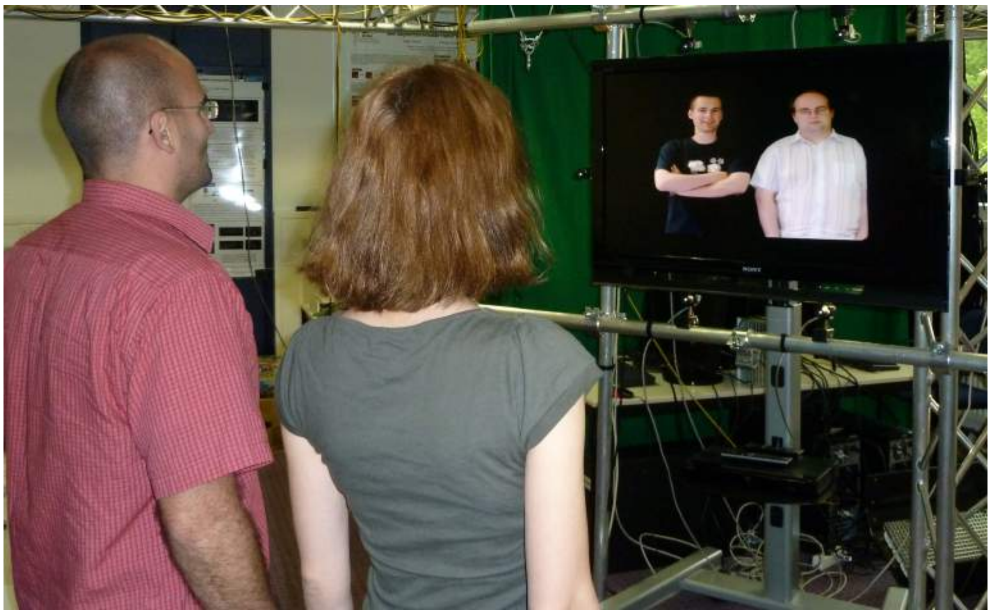


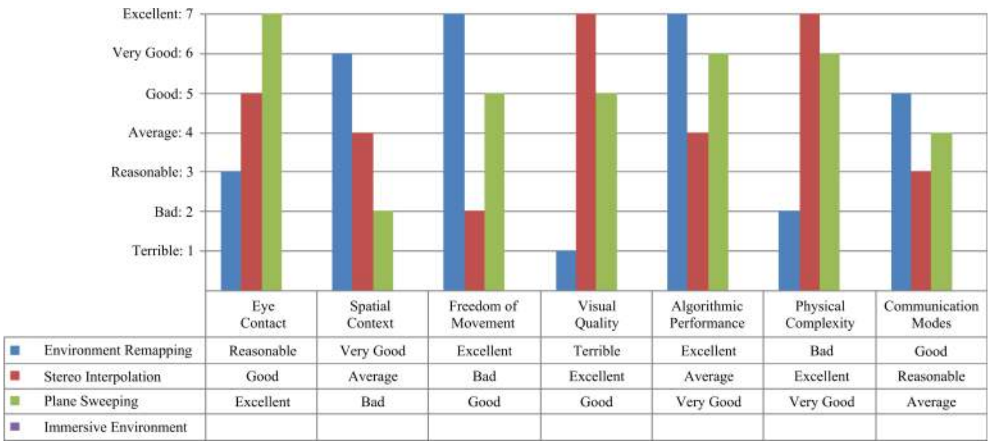
**Figure 7.18:** Our prototype for eye gaze correction using six cameras mounted around a larger screen supports limited many-to-many communication. However, a larger user-to-screen distance is required to alleviate occlusions.

**Eye Contact** The eyes are clearly visible and the eye gazes of both users are automatically locked on to each other by a concurrently running eye tracker. Moreover, the plane sweeping algorithm allows the virtual camera to move in any direction. This is much more flexible than the horizontal baseline restriction of stereo interpolation (chapter 6) and it means that eye contact is always maintained, even if a user moves. **7/7 (excellent)**

**Spatial Context** To overcome artifacts caused by occlusions that arise from the very small user-to-screen distance, the input images are segmented into foreground and background and only the foreground (i.e. the face) is interpolated. This improves visual quality, but removes spatial context. We will overcome this in our immersive environment in chapter 8. **2/7 (bad)**

**Freedom of Movement** Thanks to the eye tracker and the plane sweeping algorithm that can reconstruct the image for any virtual viewpoint (within reason), the user undeniably has more space available to move around in front of the screen, compared with the stereo interpolation solution (chapter 6). However, his space is still restricted by the bounding box of the 3D space that is visible to the cameras. **5/7 (good)**

**Visual Quality** The N-camera recoloring scheme repaints the refined depth map by averaging all six input colors, thereby causing a slight blur and washed out effect. The



| | Eye Contact | Spatial Context | Freedom of Movement | Visual Quality | Algorithmic Performance | Physical Complexity | Communication Modes |
|---|---|---|---|---|---|---|---|
| 🟦 Environment Remapping | Reasonable | Very Good | Excellent | Terrible | Excellent | Bad | Good |
| 🟥 Stereo Interpolation | Good | Average | Bad | Excellent | Average | Excellent | Reasonable |
| 🟩 Plane Sweeping | Excellent | Bad | Good | Good | Very Good | Very Good | Average |
| 🟪 Immersive Environment | | | | | | | |

**Figure 7.19:** Requirements evaluation of our plane sweeping prototype for eye gaze correction. The scores correspond to the scale defined in section 1.1, p. 3. Environment remapping has been evaluated in section 4.7, p. 63, and stereo interpolation in section 6.5, p. 124. The missing data for the immersive collaboration environment will be filled in after it has been developed in chapter 8.

confident-camera recoloring scheme overcomes this by picking the best color based on angular distance to the input cameras, at the risk of bringing out variations in illumination and photometric calibration. Regardless of the recoloring strategy, most artifacts are noticeable along the edges of the face, due to imperfections in the segmentation and the lack of a background image. **5/7 (good)**

**Algorithmic Performance** By matching multiple cameras simultaneously, the reliability of a pixel-to-pixel match is significantly increased and the need for very expensive matching cost aggregation is eliminated. This is in stark contrast to our stereo matching algorithm (chapter 5), where the cost aggregation demanded the majority of the processing time. The algorithmic complexity is further reduced by a control module that redistributes the planes in space and thereby leverages their dynamic range. The result is a fully functional prototype that confidently achieves end-to-end real-time speed on any contemporary graphics card. **6/7 (very good)**

**Physical Complexity** As with our stereo interpolation solution of chapter 6, the cameras could easily be integrated into the monitor bezel. On the other hand, we would still have to integrate multiple (six, in our prototype) instead of only two cameras. Being watched from nearby by multiple cameras might deter users. **6/7 (very good)**

**Communication Modes** The prototype has been specifically developed with close-up one-to-one communication in mind. However, it can also support limited many-to-many communication, given a large enough screen as in Figure 7.18. **4/7 (average)**

## 7.6 Conclusion

We developed a fully functional end-to-end prototype for close-up one-to-one eye gaze corrected video conferencing.

We interpolated the eye gaze corrected image from a convenient six-fold camera setup that closely surrounds the screen and avoids large occlusions. The flexible plane sweeping algorithm allows us to synthesize high-quality images from any freely selectable viewpoint, without the need of image extrapolation. Combined with a concurrently running eye tracker to position the desired viewpoint, this ensures that eye contact is maintained at all times and from any position and angle.

Using a histogram-based analysis on the depth map, the complexity control module keeps a narrow depth range focused around the user's head while he moves. This allows us to condense planes in a dynamically shifting range, instead of sweeping the entire space with a sparser distribution. This not only leverages the algorithmic performance, but also implicitly increases the accuracy of the plane sweep by significantly reducing the chance at mismatches.

Our software framework harnesses the computational resources of the graphics hardware. By exploiting the traditional graphics rendering pipeline to trick the GPU into performing general-purpose computations, we are able to achieve over real-time performance. We reach a comfortable real-time frame rate of 42 FPS for SVGA ($800 \times 600$) image resolution on an NVIDIA GeForce 8800 GTX, but our experimental setup is limited by a 30 Hz refresh rate of the cameras. We improved the end-to-end performance of the system by maximizing arithmetic intensity and by introducing granularity-based optimization schemes, without noticeable loss of visual quality. A fine-tuned set of generalized user-independent parameters allows us to synthesize eye gaze corrected images with subjectively high visual quality – rather than being geometrically correct – under variable conditions.

Our system has a minimal amount of constraints, is intuitive to use and is very convincing as a proof-of-concept.

### 7.6.1 Future Work

The geometric accuracy of the depth map could be improved by taking into account spatial and temporal constraints as explored by Kang et al. [2001] or visibility constraints following the fusion principles of Merrell et al. [2007].

The confident-camera recoloring scheme could be improved by incorporating visibility and resolution constraints in a camera blending field, similar to unstructured lumigraph rendering [Buehler et al., 2001]. Alternatively, techniques from floating textures to color the geometry proxy could be investigated [Eisemann et al., 2008].

In the largest extent, the background could be interpolated with correct motion parallax. This would complete the immersive effect of a virtual window into the remote user's world.

# Chapter 8

# Immersive Collaboration Environment

## Contents

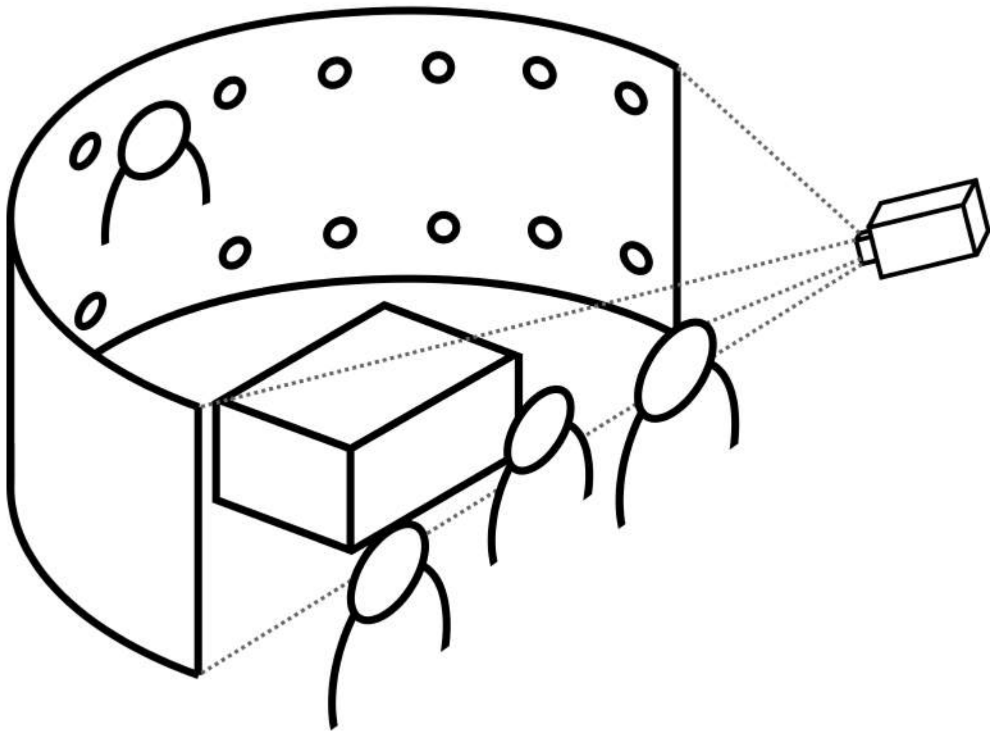


**Figure 8.1:** This chapter designs an immersive collaboration environment that corresponds to the many-camera solution depicted in the overview in Figure 1.1(d), p. 2.

We ultimately arrive at the goal that we initially set in the introduction in chapter 1: the realization of a spatially immersive environment that enables computer-supported cooperative work (CSCW) [Schmidt and Bannon, 1992] and video conferencing with restored eye gaze. Professional collaboration over large distance has and continues to become more and more a necessity. This creates the need for an immersive environment where people can instantly communicate and collaborate at a distance, as if they were in the same place at the same time, anywhere and anytime. Envisioned by Raskar et al. [1998] as *The Office Of The Future* in Figure 8.2, it brings together several fundamental areas of computer science, not in the least a unification of computer vision and graphics. Three fundamental components emerged from Raskar's work to crystallize this idea:

(1) **Dynamic Image-Based Modeling**: The computer vision part that acquires and analyzes the scene and potential display surfaces and dynamically models them in real-time for further processing or augmentation with virtual images.

(2) **Rendering**: The computer graphics part that renders the models according to the possibly irregularly shaped display surfaces, position and viewing direction of the users.

(3) **Spatially Immersive Displays**: The hardware part that provides a sufficiently immersive medium for displaying the virtual or augmented images of the generated models and physical scene. The display should at least engulf the user's peripheral vision.

These system components are enumerated independently from the description of their actual implementation and it is our goal in this chapter to specify such an implementation [Dumont et al., 2010, 2011]. Although this of course does not exclude the use of many of the techniques others have developed in each of these areas, it remains a challenge because it requires the integration of advanced computer vision, graphics and hardware that must all function seamlessly together within a real-time networked constraint.

This chapter presents our fourth and final prototype that we originally introduced in the overview in Figure 1.1(d), p. 2 (repeated in Figure 8.1). It is organized as follows. First, we discuss background and related work on immersive video conferencing and collaboration environments in section 8.1. Then, how we choose to implement – and extend – the fundamental components is described in section 8.2. The image-based modeling (i.e. component (1)) will require the plane sweeping algorithm from chapter 7. Therefore, in section 8.3 a histogram-guided approach is presented (building on the approach from section 7.2.5, p. 142) to increase the efficiency of the plane sweep for complex scenes with multiple subjects in view. Prototype results are presented in section 8.4, after which we have the familiar discussion on the requirements in section 8.5. Finally, we conclude in section 8.6.

The reader is encouraged to recognize that many key research areas that have been the focus of our research group as a whole over the past years are truly brought together: view interpolation for free viewpoint video, calibration of camera networks, tracking, omnidirectional cameras, multi-projector immersive displays, multi-touch interfaces, and audio processing.

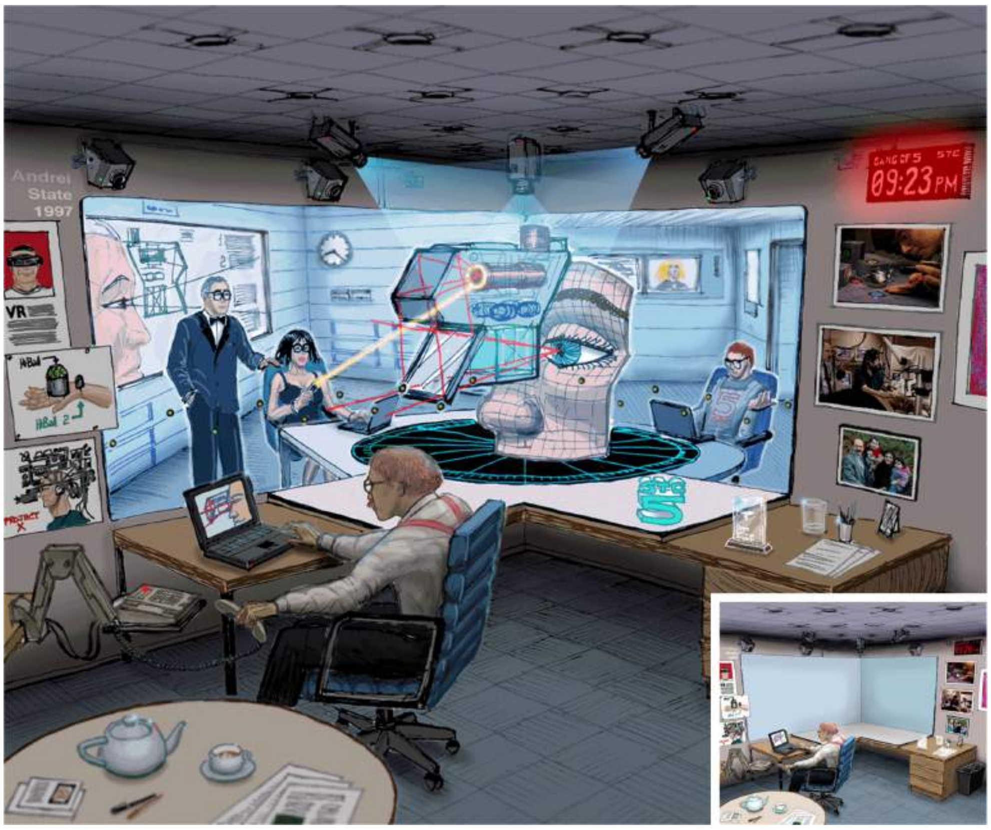


**Figure 8.2:** *The Office Of The Future* as envisioned in a conceptual sketch by Raskar et al. [1998]. The inset helps to differentiate between projected images and real objects.

## 8.1 Related Work

Cruz-Neira et al. [1993] introduced the concept of an environment that completely immerses the user in a virtual world and responds to his movement. Their CAVE (Cave Automatic Virtual Environment) was build as a projection-based surround-screen with six walls in the form of a cube. It was conceived as a virtual reality tool for scientific visualization and therefore had a sole focus on being a spatially immersive display without support for collaboration at a distance. Interaction between the installation and the virtual world was supported via wired head and hand trackers, while stereo glasses provided passive stereoscopic 3D.

Later, Green and Whites [2000] identified the components of the CAVE design that represent the major costs and redesigned them with the aim of reducing cost for the end-user as much as possible, while preserving most of the CAVE's functionality. Named the CAVE-let, they imagined a wide range of applications in areas such as tele-health, visualization, education and collaboration. By purposing the CryEngine2 video game engine, Juarez et al. [2010] also implement a low-cost CAVE-like environment and give it the moniker CryVE.

Acknowledging that CAVE-like environments had played a minor role so far in advancing the sense of immersion for video conferencing systems, Childers et al. [2000] explored their use as the basis for building immersive visualization and collaboration environments. In their Access Grid project, they learned that a strong sense of immersion and the provision of natural sight and audio cues is important to facilitate collaboration between users at remote sites. They make passing mention of, but do not offer any tangible solution to, the specific problem of supporting eye contact.

The Blue-C project [Gross et al., 2003] is another pioneer of CAVE-like environments, intended for virtual design and collaboration. It consists of three glass panels (front, left and right) that contain liquid crystal layers. By varying electric current, these layers can be switched from a whitish opaque state (for projection) to a transparent state (for acquisition). The opaque state allows the panels to be projected on by two projectors in an active stereo configuration, while the transparent state allows the video cameras to look through the panels and capture the user. From the captured user video streams, a 3D model is reconstructed and integrated into a networked virtual environment at the remote site. Eye contact is implicitly approximated by cameras positioned directly behind the transparent screens. However, switching between opaque and transparent states demands precise synchronization by special-purpose micro-controller hardware. Additional hard- and software components accomplish head tracking, spatial audio, voice communication and user interaction.

Moving away from fully encapsulating CAVE-like environments, the most common approach to realize immersive collaboration is through the concept of a shared virtual table environment (SVTE). The physical and virtual environment is set up in such a way that all users have the impression of sitting around a shared table and being part of an extended space, thereby enabling a higher degree of natural interaction and collaboration.

Early implementations of the SVTE resulted in tele-cubicles, where all users are seated symmetrically around the virtual table and each user appears on his own dedicated window-like screen [Aoki et al., 1999; Gibbs et al., 1999; Chen et al., 2000]. This specific arrangement of single-user terminals restricts the sense of physical presence, is only suited for a fixed number of users and does not scale well.

A lot of research regarding STVEs has been performed in the VIRTUE (virtual team user environment) and 3DPresence projects of Schreer and Sheppard [2000] and Feldmann et al. [2009b]. Their goal was to offer multiple users a seamless transition between the real conference table in front of the display and the virtual table on the screen by rendering the remote users under correct perspective view into the conference scene [Schreer et al., 2001, 2008b; Kauff and Schreer, 2002; Feldmann et al., 2010]. Just like our prototypes, their system consists of an extensive processing chain that integrates rectification and lens distortion correction, audio processing, networking, segmentation, scene depth reconstruction (by stereo matching), visual hull modeling of the users and subsequent novel view synthesis to correct eye gaze, head tracking, and composition into the shared scene. Nevertheless, their shared conference scene remains virtual and not real, thus offering little spatial context and preventing the local users from receiving any information on the remote users' actual environment.

Commercially available but expensive CAVE-like systems include Barco's range of immersive displays, 3D video walls and dome displays [Barco NV] and Holovis' MotionDome for theme park entertainment (e.g. interactive rides) [Holovis, 2013]. Commercially available and equally expensive video conferencing systems include Cisco's telepresence solutions [Cisco Systems, 2009] and Polycom's RealPresence Immersive Studio [Polycom, 2009]. They do not actively correct eye gaze but rather rely on a sufficiently large user-to-screen distance.

Otto et al. [2006] and Wolff et al. [2007] state that current tools for computer-supported cooperative work (CSCW) [Schmidt and Bannon, 1992] suffer from two major deficiencies. First, they do not allow to observe the body language, facial expressions and spatial context of the collaborators. Second, they miss the ability to naturally and synchronously manipulate objects in a shared environment. To solve these issues, they suggest that only through an immersive collaborative virtual environment (CVE) can we achieve the seamless collaboration that so naturally exists in a conventional face to face meeting. Furthermore, Divorra et al. [2010] also observe that telepresence spaces need more immersive and intuitive interaction with documents and applications for more natural telecollaboration and task sharing.

In all the aforementioned systems, however, the focus still lies primarily on providing eye gaze correction and not on collaboration. We will develop an immersive collaboration environment that combines our view interpolation for eye gaze correction with a spatially immersive multi-projector display and a multi-touch surface that allows cooperation through a networked shared interface. Our environment implements and extends the technical requirements identified by Raskar et al. [1998]: dynamic image-based modeling, rendering and a spatially immersive display. Similar to our needs, Isgro et al. [2004] also identify calibration, multiple view analysis, tracking and view synthesis as the fundamental image processing modules.

## 8.2 Fundamental Components

We implement the three fundamental components previously defined in the introduction as follows. A sea of cameras and projectors lies at the base of our reconstruction and rendering algorithm that (1) dynamically models and (2) renders the environment, i.e. both the users and their background scene. Furthermore, we have created (3) a spatially immersive display on which a multi-projector setup projects an augmented reality to form a virtual extension of the physical office. Going beyond the original concept, we add (4) cooperative surface computing and (5) audio processing for realistic communication. We integrate these five components into one prototype setup, represented in Figure 8.3. We will now elaborate in detail on each of the components.

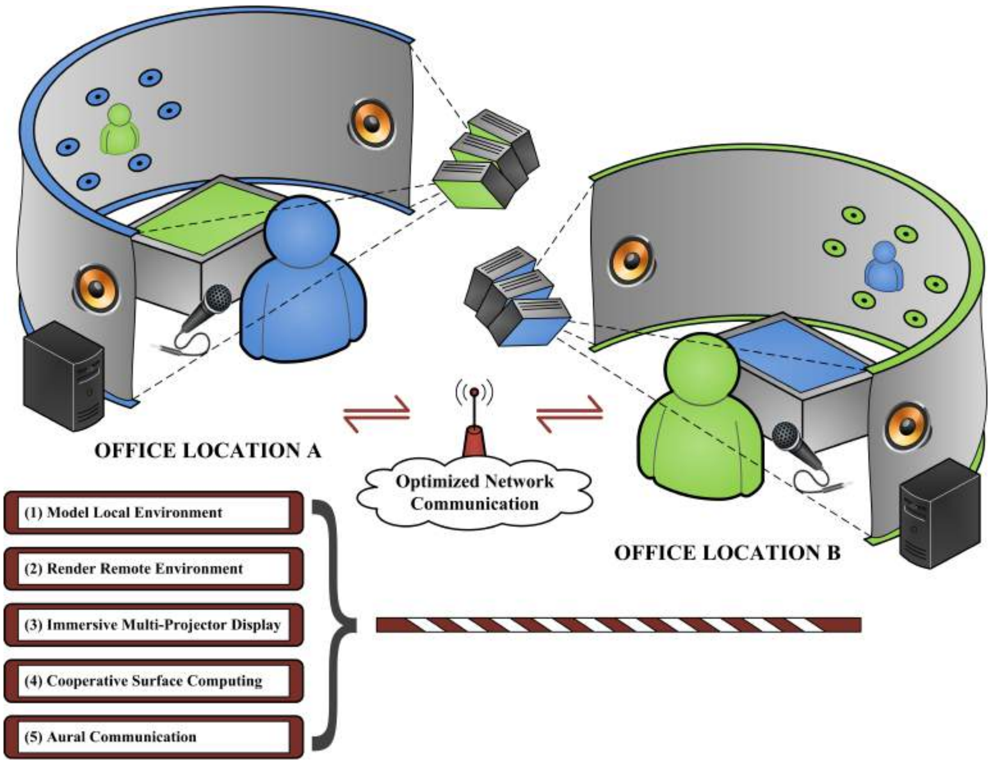

OFFICE LOCATION A
OFFICE LOCATION B
Optimized Network Communication
(1) Model Local Environment
(2) Render Remote Environment
(3) Immersive Multi-Projector Display
(4) Cooperative Surface Computing
(5) Aural Communication

**Figure 8.3:** Our immersive collaboration environment exists of five fundamental components: the ability to both (1) model the local environment and (2) render the remote environment, (3) an immersive multi-projector display, (4) cooperative surface computing and (5) aural communication.

### 8.2.1 Model Local Environment

To capture and dynamically model the local user, we make use of a sea of cameras that are placed behind the panoramic screen. Small holes are cut in the screen that allow the lenses to peek through, while at the same time being spaced far enough apart so as to not interfere with the user experience of being immersed in the environment. Consequently, this calls again for the application of view interpolation to correct the user's eye gaze and we now have a choice between either rectified stereo from chapters 5 & 6 or plane sweeping from chapter 7. The curved nature of the panoramic screen, however, implies that the cameras' centers of projection do not lie in the same plane and thus are certainly not rectified with respect to one another. This makes it less feasible to opt for stereo interpolation and all the more feasible to opt for the plane sweeping algorithm, which inherently takes into account a more arbitrary camera configuration anyway. In fact, our plane sweeping approach from chapter 7 can be carried over in its entirety and applied with minimal modifications, which in itself is great proof of its flexibility.

In practice, the cameras are laid out similarly to the configuration used in chapter 7, which means we limit their number to six pieces. However, because they are now spaced further apart and because of a larger user-to-cameras distance, we are able to reconstruct a much wider view of the user. That is to say, we can include the user's torso and arms and offer him a larger freedom of movement (e.g. he can move his arms, can sidestep, etc.), without impeding on the resulting interpolation quality.

One limitation of applying plane sweeping as developed in chapter 7, is the fact that it completely discards the background of the user; take a look back at Figure 7.15, p. 151, for example. This was originally welcomed as a means to enhance the quality of the generated virtual imagery, but it now poses a problem if our goal is to fully immerse the users in each other's environment. Our solution is to capture the local user's panoramic background using a separate omnidirectional camera positioned behind him and subsequently project it on the panoramic screen of the remote user. Specifically, we use the Point Grey Ladybug3 omnidirectional camera shown in Figure 8.4(b). Optionally, a stereo rig could even be brought into play, together with view interpolation algorithms, to enable parallax effects at the rendering side. Omnidirectional cameras that directly capture stereoscopic panoramas for display on a cylindrical screen are also available [Peleg et al., 2001; Weissig et al., 2012].

Another limitation that requires our attention is the fact that the spatial nature of our system (large panoramic screen, lots of space to move around) could easily support multiple users together, occupying the same immersive environment. However, the complexity control module as developed for the plane sweeping algorithm in section 7.2.5, p. 142, supports only one user at the same time, two at the most if side-by-side (as in Figure 7.18, p. 155). We will tackle this problem in section 8.3.

Lastly, in an environment as complex and highly dynamic as this one, it can be a challenge to effectively calibrate the cameras and even more so to keep them calibrated afterward. These challenges can be met perfectly well using the recalibration procedures presented in chapter 3.

### 8.2.2 Render Remote Environment

After the remote user and his background scene have been captured separately as described in section 8.2.1, they should now both be rendered correctly according to the viewing position and orientation of the local user.

As a first step, the background scene that has been captured at the remote site is displayed locally, after it has been geometrically corrected for projection on the panoramic screen, as will be explained in section 8.2.3. By exchanging these background scenes between users, the environments are augmented with a truly immersive dynamic scene, which allows the users to gain information about the remote environment and (ideally) infer from each other what they are gazing at. In contrast, in their original conception of *The Office Of The Future*, Raskar et al. [1998] scan the surrounding background environment off-line (using a laser range finder) and render it as a static scene.

On top of the background layer, the remote user is rendered from the correct virtual viewpoint, so that eye contact between both collaborators is restored. To achieve this, we must track the local user's eyes, so that the coordinates can be sent over the network to the remote site, where they can guide the plane sweeping algorithm that models the user to only reconstruct the required point of view. In essence, this is completely analogous to the system flow described in section 7.2.6, p. 145, and the same eye tracking module can be employed. However, because the user-cameras distance is now much larger, less pixels are available to capture the eyes and hence the tracking will be more error-prone. We counter this by introducing a two-step hierarchical algorithm, where first the visible body of the user is tracked and only then the coordinates of his eyes inside his head are estimated [Maesen, 2016].

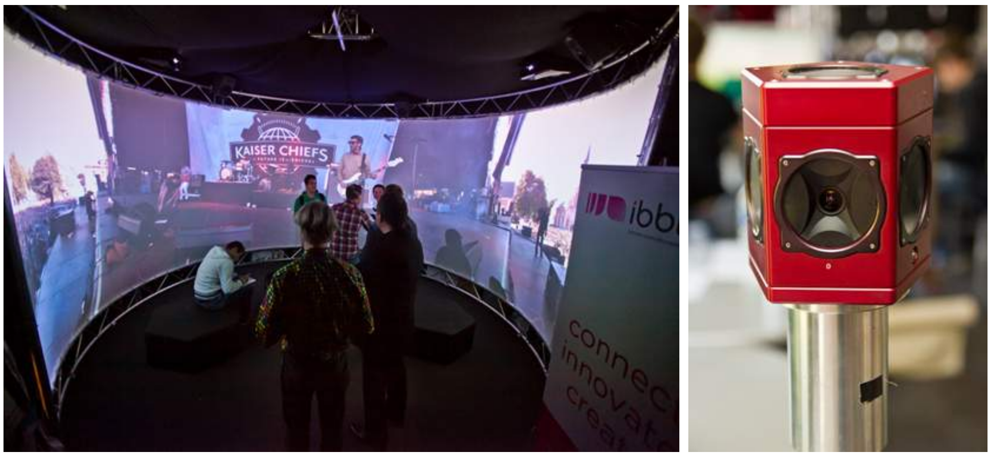
In case the background scene has been captured by a stereo rig in section 8.2.1, the same eye coordinates can guide the interpolation between the left and right stereo viewpoint (handled by the algorithms of chapters 5 & 6) and thereby introduce the correct motion parallax when the user moves his head side to side.

Although all of this combined already results in an undeniably high sense of immersivity, it can be even further improved by optionally rendering both the background and remote user stereoscopically for natural 3D perception [Dumont et al., 2009a; Rogmans et al., 2010a]. However, as this would require either an autostereoscopic panoramic screen (something that would at best be financially unavailable to all but a select few users) or the inconvenience of wearing active shutter glasses to perceive the 3D effect (which would destroy eye contact), we resort to exploiting only monocular depth cues (e.g. a user that is further away is rendered smaller) while still perceiving good 3D [Held et al., 2010; Rogmans et al., 2010b].

### 8.2.3 Immersive Multi-Projector Display

The third fundamental requisite of a futuristic office is a spatially immersive display that at least engulfs the user's peripheral vision. We designed an easy to build and very affordable 180-degree panoramic screen by spanning a durable white vinyl cloth over a truss constructed of lightweight aluminum bars. To suppress the overall cost, the cloth is matte white with reflection less than 5%, instead of the cinematic pearlescent screens with a reflection of about 15%. It is shown in Figure 8.4(a). The full extent of the screen is projected on by multiple synchronized projectors that are placed above and behind the user, projecting over his head at a slightly downward angle.

The multi-projector setup must be calibrated to be able to seamlessly stitch the multiple projections together. More specifically, the projected image must be geometrically aligned to compensate for the irregular shape of the screen and the intensity of overlapping projector pixels must be adjusted. While various methods exist [Li and Chen, 1999; Raskar et al.,



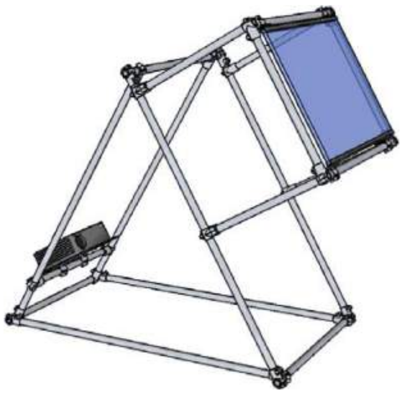
(a) Panoramic screen. (b) Omnidirectional camera.

**Figure 8.4:** (a) Our panoramic screen displaying a stage performance by Kaiser Chiefs, (b) recorded using the Point Grey Ladybug3 omnidirectional camera.

1999; Fiala, 2005; Weissig et al., 2005; Harville et al., 2006; Griesser and Van Gool, 2006; Sajadi and Majumder, 2010], our internally developed multi-projector calibration procedure is based on perceptible structured light Bekaert [2008–2012]. The same omnidirectional camera that captures the user's background (shown in Figure 8.4(b)) is temporarily placed in the *sweet spot* of the environment, i.e. the location for which a perfect perspective view is created regardless of the actual shape of the screen. Consequently, instead of stitching multiple cameras from multiple viewpoints together, we need only to perform only one omnidirectional capture of the structured light pattern. Finally, the capture is synchronized temporally for varying structured light patterns and requires only mere seconds to complete. The method produces visually seamless images, even on screens that are not perfectly cylindrical or spherical. The calibration is fully automatic and requires no user intervention, which makes it ideal for casual setups such as immersive home video conferencing systems.

### 8.2.4 Cooperative Surface Computing

Beyond the three hitherto defined fundamental components to build an office of the future, we also provide collaboration through networked multi-touch surface computing [Dietz and Leigh, 2001; Wobbrock et al., 2009]. Although this is not a direct criterion in the original draft of *The Office Of The Future* [Raskar et al., 1998], it greatly contributes to the collaborative characteristics and we deem it to be an essential part of our immersive collaboration environment. Internally built, our solution consists of both a hardware system [Cuypers et al., 2008; TinkerTouch, 2010] and accompanying software framework [Cuypers et al., 2009].


(a) Development sketch.

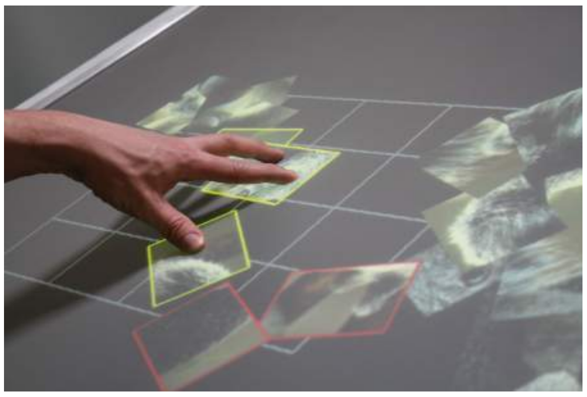

(b) Networked cooperative jigsaw puzzle.

**Figure 8.5:** (a) Sketch of our internally built multi-touch surface, based on frustrated total internal reflection of infrared light [Han, 2005]. (b) A cooperative jigsaw puzzle game running on two networked multi-touch surfaces. A yellow border indicates that a puzzle piece is being manipulated by the local user, while a red border marks a piece that is concurrently being manipulated by the remote collaborator.

The hardware is based on work by Han [2005], who makes use of frustrated total internal reflection (FTIR) of infrared light. The screen consists of an acrylic plate in which infrared light is injected. On top of the acrylic is a compliant surface film and a rear-projection film. Touching the projection film will cause optical contact between the compliant surface and the acrylic, resulting in local scattering of the infrared light. An infrared camera placed below the surface (together with a projector) is able to detect the scattered light and hence locate multiple touch points, which in turn allows an (theoretically) unlimited amount of users to interact on the screen at the same time. A development sketch is shown in Figure 8.5(a).

On the software side, a framework is provided that allows files to be shared over the network and consequently be viewed, controlled, manipulated and annotated by both sides simultaneously, as if all users were actually working at the same surface. Applications that were available at the time of development of the prototype include a photo and video browser, a (pdf) document browser and a cooperative jigsaw puzzle game. The latter can be observed in action in Figure 8.5(b). Note the yellow border around puzzle pieces that are being manipulated by the local user and the red border around pieces that are concurrently being manipulated by the remote collaborator.

### 8.2.5 Aural Communication

Finally, we include advanced audio processing to further facilitate and complete the communication between the users. For now, we only offer support for monaural sound that is captured with a single high-fidelity microphone and subsequently sent to the remote site for processing. Upon arrival, the audio stream is first registered and synchronized before being amplified and outputted through the speakers.

The Larsen effect, i.e. the audio feedback that is generated through the loop between both audio systems, is canceled on-the-fly using the network transfer delay determined at registration. The required audio processing is implemented in Pure Data (Pd) [Puckette, 1996], an open source visual programming language that allows to create and implement processing chains to not only generate audio, but also video, graphics, interface sensors, input devices and MIDI. For echo cancellation we use the normalized least mean square (NLMS) adaptive algorithm to update the filter coefficients iteratively to minimize the error between the output and desired signal [Holzmann and Strobl, 2005].

The synchronization and echo cancellation contribute to a natural way of communicating, giving the users the feeling of talking to each other in the same room. Although not yet implemented, the system design lends itself perfectly for localized 3D audio by using a minimum of two microphones and reconstructing the 3D sound for playback by a surround speaker setup. This also provides the users the direction of the speaker, which is particularly useful during many-to-many collaboration scenarios. However, the sound must remain consistent with the augmented reality.

## 8.3 Complexity Control: Adaptive Non-Uniform Plane Distribution

Our prototype of chapter 7 employed a plane sweeping algorithm that places $M \geq 2$ planes at uniformly distributed depths $\{z_1, \ldots, z_j, \ldots, z_M\}$ between a nearest and farthest depth $z_{min}$ and $z_{max}$. Each depth $z_j$ is determined as:

$$z_j = z_{min} + \sigma_j (z_{max} - z_{min}) \tag{8.1}$$

where $\sigma_j \in [0, 1]$ is a normalized depth value that uniformly distributes the planes as:

$$\sigma_j = \frac{j-1}{M-1} \tag{8.2}$$

so that $z_1 = z_{min}$ and $z_M = z_{max}$ (cf. Equation 7.4, p. 135).

Uniformly distributed planes, however, may be positioned at depths where no objects are actually present in the scene. The chance of this happening is especially high in scenes that are only sparsely populated with objects, as demonstrated in Figure 8.6(left). This not only wastes a lot of computational resources, but also increases the risk at artifacts caused by mismatches.

In section 7.2.5, p. 142, our solution was to dynamically adjust the nearest and farthest plane depths, guided by a Gaussian fit on the histogram of the depth map. As explained in Figure 7.10, p. 144, this effectively kept the depth range focused around the user at all

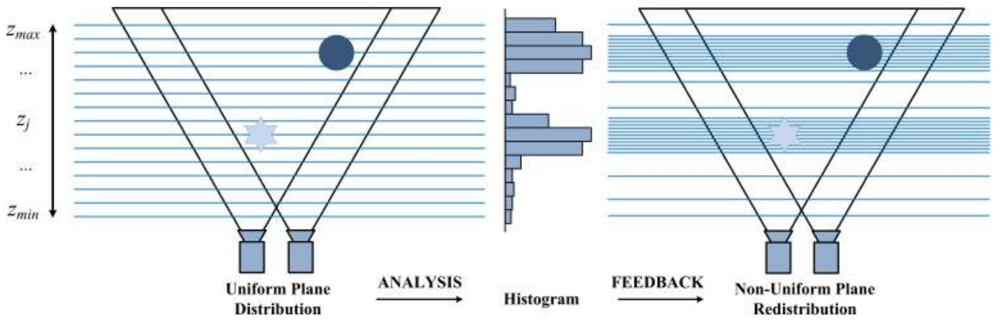


**Figure 8.6:** (left) Uniformly distributed planes may be positioned at depths where no objects are actually present in the scene. (center) Peaks in the histogram of the depth map indicate at what depth objects are located in the scene. (right) Guided by the histogram, the planes can be rearranged into a non-uniform distribution that provides less planes in ranges with less objects and more planes in ranges with more objects.

times. Although this approach proved very powerful for one-to-one video conferencing, it quickly shows its limitations when multiple users are present at too differing depths in front of the cameras. As this could exactly be the case in our immersive collaboration environment, we here develop a method to adapt the distribution of the planes to the actual scene content [Goorts et al., 2013b; Goorts, 2014; Goorts et al., 2014b; Dumont et al., 2014a].

To redistribute the planes according to the scene content, we must again appeal to the depth map histogram. As shown in Figure 8.6(center), peaks in the histogram indicate at what depth objects are located in the scene. We can interpret this histogram as a probability density function that describes the likelihood that a plane should be positioned at a particular depth in the scene. In other words, the planes are rearranged to provide less planes in ranges with less objects and more planes in ranges with more objects. This results in the non-uniform plane distribution shown in Figure 8.6(right). We can express this non-uniform distribution mathematically as:

$$z_j = z_{min} + \zeta_j (z_{max} - z_{min}) \tag{8.3}$$

where the purpose of $\zeta_j$ is to map a plane number $1 \leq j \leq M$ to a non-uniform distribution in the normalized depth range $[0,1]$, similar to how $\sigma_j$ of Equation 8.2 generates a uniform distribution. Our goal now becomes to define $\zeta_j$ for each plane number $j$.

We begin by constructing the histogram $H(j)$ of the depth map of the current frame, represented in Figure 8.7(left). Because the current depth map may itself be the result of a non-uniform plane distribution $\zeta_j$ coming from the previous temporal frame, depth values $z_j$ in the histogram $H(j)$ are binned according to plane number $j$ rather than depth value $z_j$ directly. Doing so facilitates implementation by ensuring that the bin centers remain uniformly distributed. Next, we convert the histogram to its cumulative version $\bar{H}(j)$. Here, each bin

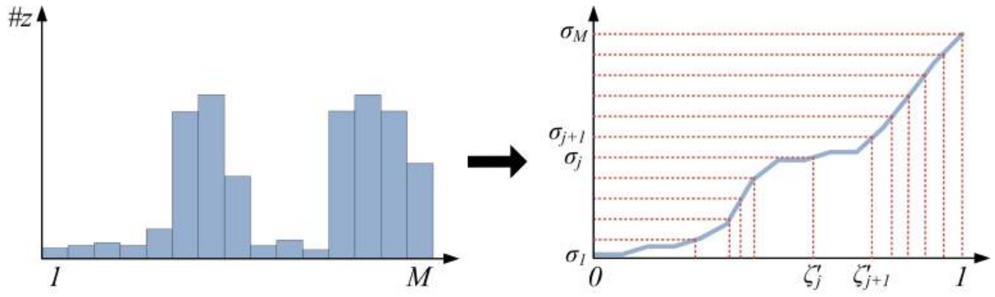


**Figure 8.7:** (left) The histogram $H(j)$ of the depth map. (right) Its corresponding cumulative histogram $\bar{H}(j)$, rescaled and interpreted as a continuous monotonically increasing function $h(x) = y$. For values of $x$ with lots of corresponding depth values in the depth map, $h(x)$ will be steep. For values of $x$ with few corresponding depth values, $h(x)$ will be flat. We can use this observation to rearrange the planes from a uniform distribution of $\sigma_j$ on the Y-axis to a non-uniform distribution of $\zeta'_j$ on the X-axis.

not only counts the number of occurrences of the specific depth value it collects, but also includes the count of all previous bins.

Following the construction of the cumulative histogram $\bar{H}(j)$, we rescale both its axes to the normalized range $[0,1]$. The bin centers $j$ on the X-axis are converted to their normalized depth value $\zeta_j$, while the Y-axis is scaled by dividing every cumulative bin count by the total number of depth values that have been binned. This provides us with $M$ samples $(x_j, y_j)$:

$$x_j = \zeta_j \tag{8.4}$$

$$y_j = {}^{\bar{H}(j)}\!/_{\bar{H}(M)} \tag{8.5}$$

The result can be interpreted as a continuous monotonically increasing function $h(x) = y$, shown in Figure 8.7(right). Its domain $x = [0,1]$ represents normalized depth, while its image $y = [0,1]$ is the normalized cumulative count of depth values. We can determine the function $h(x)$ by fitting a $(M-1)^{\text{th}}$-degree polynomial on the $M$ samples $(x_j, y_j)$:

$$h(x) = p_1 \cdot x^{M-1} + p_2 \cdot x^{M-2} + \ldots + p_{M-1} \cdot x + p_M \tag{8.6}$$

with $p_j$ its $M$ coefficients.

If we now divide the Y-axis in $M$ cross sections $\sigma_j \in [0,1]$, always uniformly distributed according to Equation 8.2, then we arrive at the desired definition of $\zeta_j$ as:

$$\zeta'_j = h^{-1}(\sigma_j) \tag{8.7}$$

where the notation $\zeta'_j$ indicates that this non-uniform distribution will be used in the next temporal frame.

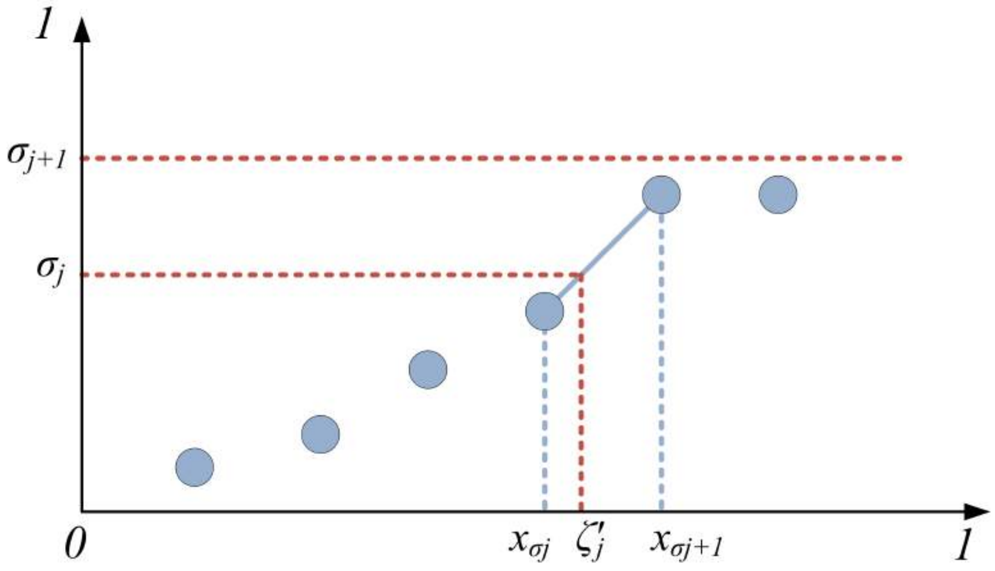


**Figure 8.8:** Detail of the rescaled cumulative histogram of Figure 8.7(right), with discrete values $(x_j, y_j)$. The non-uniformly distributed and normalized plane depth $\zeta'_j$ is linearly interpolated between the histogram bin centers $x_{\sigma j}$ and $x_{\sigma j+1}$, where the latter are determined so that $\bar{H}(x_{\sigma j}) \leq \sigma_j$ and $\bar{H}(x_{\sigma j+1}) > \sigma_j$

This definition provides a transformation from plane numbers $j$ to a uniform distribution of (what can be regarded as normalized plane numbers) $\sigma_j$ on the Y-axis and from there to a non-uniform distribution of normalized depth values $\zeta'_j$ on the X-axis. As can be verified visually by tracing the dotted lines in Figure 8.7(right), planes will be densely distributed where the cumulative histogram is steep, whereas a sparse distribution emerges where the cumulative histogram is flat.

One way to determine $h^{-1}(y)$ is to fit a $(M-1)^{\text{th}}$-degree polynomial on the samples $(y_j, x_j)$. Much more efficient and still sufficiently accurate, however, is to use piecewise linear interpolation. This is explained in Figure 8.8, which zooms in on some of the discrete samples $(x_j, y_j)$. First we determine the normalized depth value (i.e. histogram bin) $x_{\sigma j}$ for which $\bar{H}(x_{\sigma j}) \leq \sigma_j$ and $\bar{H}(x_{\sigma j+1}) > \sigma_j$. Once $x_{\sigma j}$ is determined, $\zeta'_j$ can be linearly interpolated:

$$\phi = \frac{\sigma_j - \bar{H}(x_{\sigma j})}{\bar{H}(x_{\sigma j+1}) - \bar{H}(x_{\sigma j})} \tag{8.8}$$

$$\zeta'_j = \phi x_{\sigma j+1} + (1-\phi) x_{\sigma j} \tag{8.9}$$

which can then be plugged into the plane sweeping algorithm to redistribute the planes for the next temporal frame by equating $\zeta_j = \zeta'_j$ in Equation 8.3.

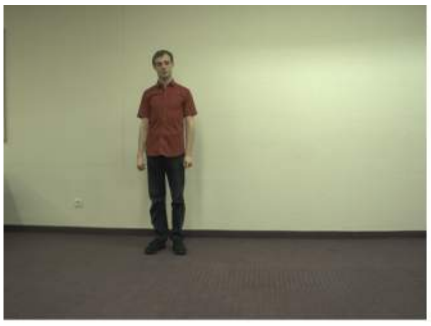
It is desirable to include some planes in the empty space between objects to allow the (re-)appearance of objects in dynamic scenes. To achieve this, a fixed value, proportional the number of pixels, is added to all bins in the histogram. This way, the cumulative histogram will be less flat in less interesting regions, resulting in some planes being assigned there. In our tests, adding 0.1% of the total amount of pixels reached the desired effect. Furthermore, for stability and to prevent collapse of the depth range, care should be taken to ensure that $z_1 = z_{min}$ and $z_M = z_{max}$ never change.

**Evaluation** First, to validate the reasoning behind the use of the cumulative histogram, Figure 8.9(a) shows an input image of a video sequence shot with a single immersive environment user in view. The presence of only one dominant depth in the scene (i.e. the single user) causes only one steep section in the cumulative histogram of the depth map, shown in Figure 8.9(b). By Equation 8.7, this steep part is transformed to a flat value of $\zeta'_j$, which can be understood from its graph in Figure 8.9(d). Conversely, flat sections of the cumulative histogram correspond to steep values in the same graph of $\zeta'_j$. The resulting non-uniform plane distribution is visually represented in Figure 8.9(c), with an appropriately increased or decreased density of planes in the corresponding regions of the depth range.

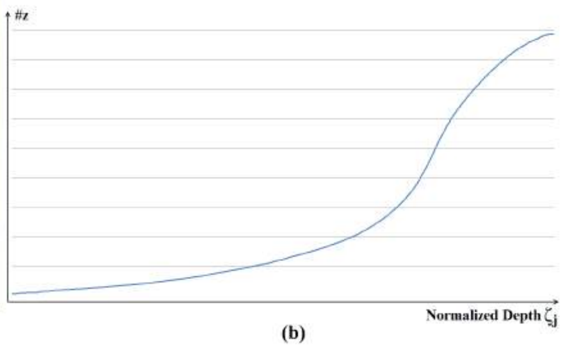
In Figure 8.9(e) a second user joins the party, so that now the scene contains multiple dominant depths. In turn, the cumulative histogram in Figure 8.9(f) also contains multiple steep sections. This ultimately results in multiple dense regions in the plane distribution, as reflected by the values of $\zeta'_j$ in Figure 8.9(h) and their visual representation in Figure 8.9(g).

For one more scene with multiple users in Figure 8.10, we compute the depth map with a varying amount of both uniformly and non-uniformly distributed planes. Figure 8.10(b) shows the result for a uniform distribution of a low number of planes (50). The depth map clearly contains artifacts, caused by the inherently sparse plane distribution. After non-uniformly redistributing the planes, the depth map in Figure 8.10(d) shows less noise and outliers. Furthermore, the silhouettes are delineated more finely and the features of the users are more discernible. The artifacts are effectively filtered out by the non-uniform plane distribution, as the planes that generate them are excluded from contributing to the depth map. For comparison, Figure 8.10(c) shows the result for a high number of uniformly distributed planes (256). Here, some noise and vague edges are still present. As the non-uniform plane distribution generates an improved depth map by still using the same low number of planes, the overall system performance increases significantly.

**Figure 8.9** ***(facing page)*****:** Non-uniform plane distribution for (a)–(d) one and (e)–(h) two users. (a),(e) An input image of the scene. (b),(f) The cumulative histogram $\bar{H}(j)$ of the depth map, rescaled to the continuous fuction $h(x) = y$ of Equation 8.6. (d),(h) The corresponding $\zeta'_j$ for each given normalized plane number $\sigma_j$, as determined by the inverse function $h^{-1}$ of Equation 8.7. (c),(g) A visual representation of the resulting non-uniform plane distribution.

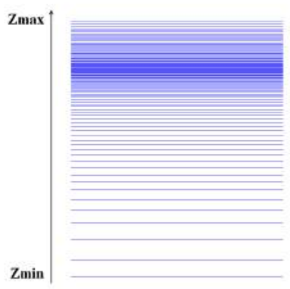


(a)



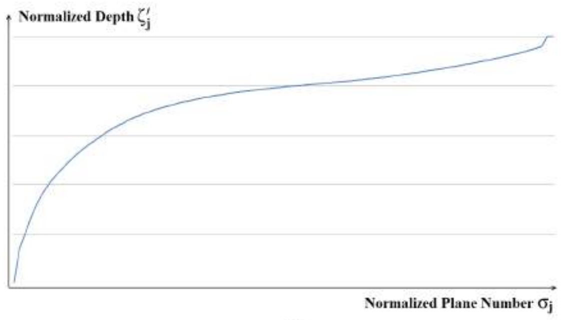
#z
Normalized Depth $\zeta_j$

(b)



Zmax
Zmin

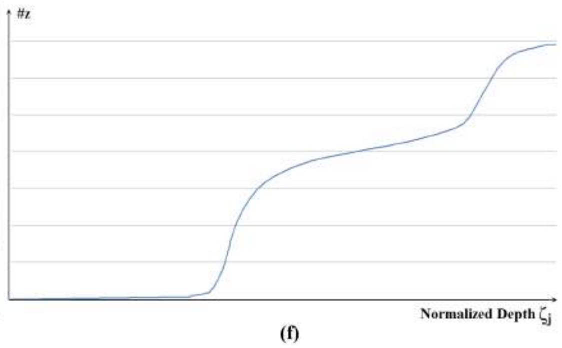
(c)



Normalized Depth $\zeta'_j$
Normalized Plane Number $\sigma_j$

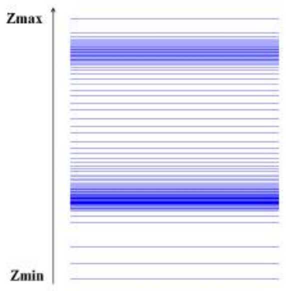
(d)

(e)



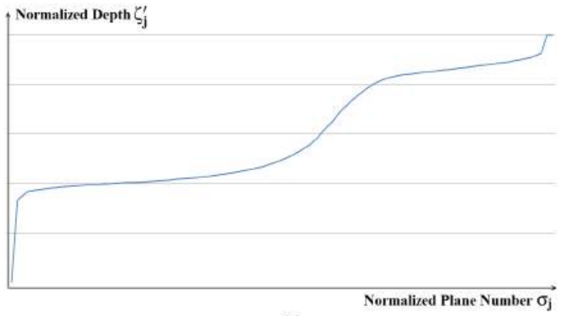
#z
Normalized Depth $\zeta_j$

(f)



Zmax
Zmin

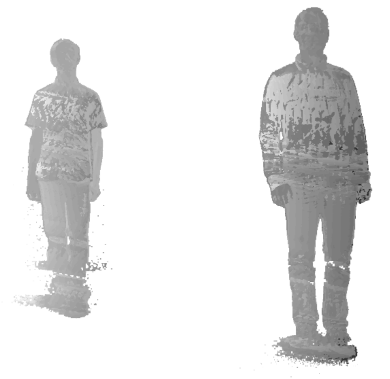
(g)

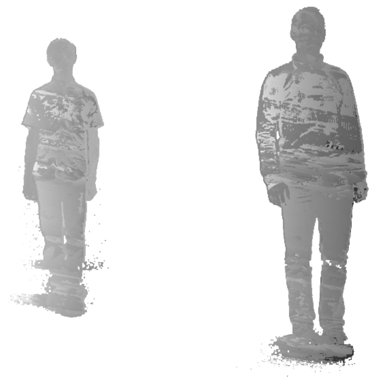


Normalized Depth $\zeta'_j$
Normalized Plane Number $\sigma_j$

(h)

(a) Input image.


(b) Uniform distribution of
a low number of planes (50).


(c) Uniform distribution of
a high number of planes (256).

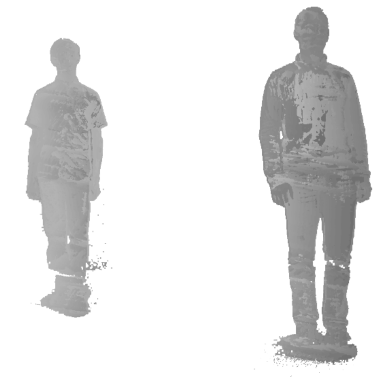

(d) Non-uniform distribution of
a low number of planes (50).

**Figure 8.10:** Comparison of depth maps computed with a uniform versus a non-uniform plane distribution and a low versus a high number of planes. (a) Input image of a scene with two users. (d) The depth map generated by a non-uniform plane distribution, based on the histogram of (b) the depth map generated by a uniform plane distribution, contains less noise and outliers in its depth values. Nonetheless, the non-uniform plane distribution still uses the same low number of planes, which improves the overall system performance significantly. (c) In comparison, simply increasing the number of planes of the uniform distribution does not improve the quality as much.

So far we have always used the depth map of the previous temporal frame to determine the plane distribution for the next temporal frame. This may affect the quality of the depth map in highly dynamic scenes in particular, as the plane distribution for the next iteration is determined only at the end of the current iteration. One way to counteract this is to ensure

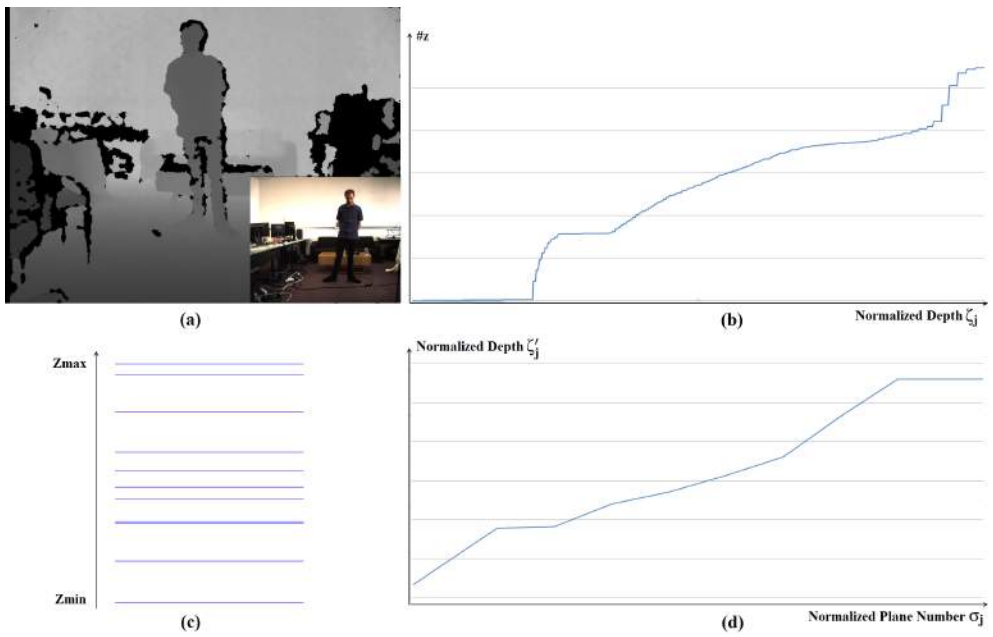
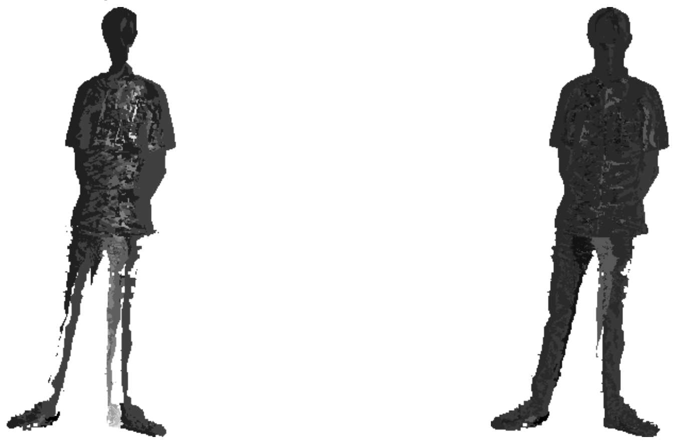


**Figure 8.11:** Non-uniform plane distribution guided by an active depth camera, in this case the Microsoft Kinect [Zhang, 2012]. The depth map that the Kinect captures is of lower quality and does not suffice for view interpolation, but can still provide a cumulative histogram that is sufficiently accurate to guide the redistribution of the planes. (a) The depth map and color image (inset), both captured by the Kinect. (b)–(d) As in Figure 8.9.



(a) Uniform distribution. (b) Kinect-guided non-uniform distribution.

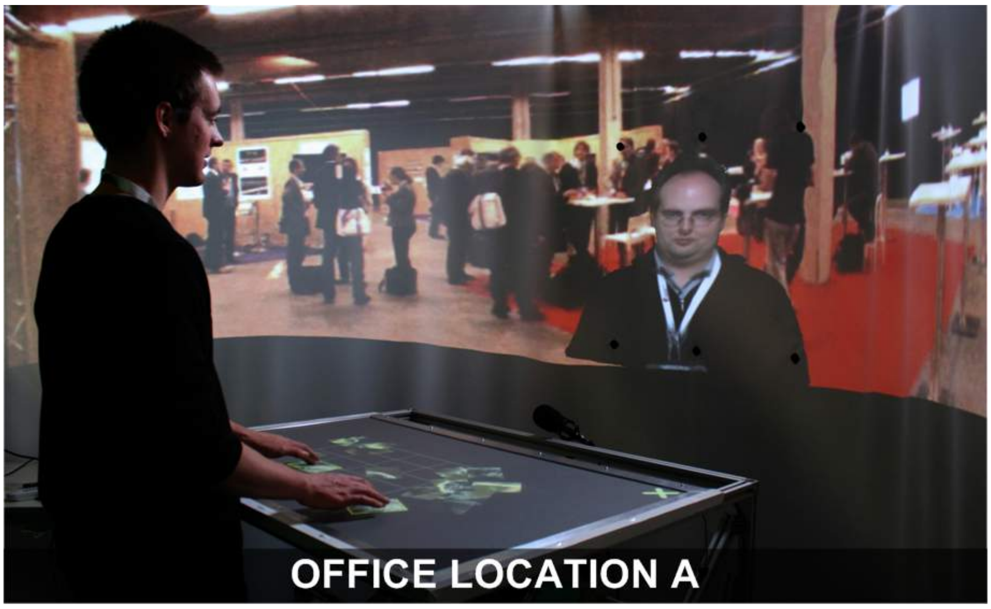
**Figure 8.12:** Comparison of the depth maps computed with (a) a uniform versus (b) a non-uniform plane distribution. The non-uniform distribution of the planes has been guided by the depth map captured by a Kinect. Consequently, only a low number of planes (10) is required.

high system responsiveness, i.e. a high processing frame rate, as we did for the shifting depth range in section 7.2.5, p. 142. Another solution is to rely on active depth cameras to capture an initial coarser depth map of the scene. Our final experiment demonstrates the use of a Microsoft Kinect [Zhang, 2012] to obtain an initial depth map and subsequent histogram, with the aid of which the planes are then redistributed in the current frame. The depth map that the Kinect captures is of lower quality and therefore does not suffice for view interpolation, as can be observed in Figure 8.11(a). The Kinect depth map can, however, still provide a cumulative histogram (Figure 8.11(b)) that is sufficiently accurate to guide the redistribution of the planes. As an added advantage, less planes are required in empty spaces to compensate for moving objects, because an actively captured depth map will inherently contain those objects anyway. The resulting non-uniform plane distribution is visualized in Figure 8.11(c)–(d). Next, Figure 8.12(a) shows the result for a uniform distribution of a low number of planes (10). The depth map contains lots of errors, noise and artifacts, and many parts are missing or incorrect. In contrast, Figure 8.12(b) shows the result for a non-uniform distribution of the same low number of planes, where the distribution is based on the depth map provided by the Kinect. This depth map is much more complete. Noise and artifacts are greatly reduced, while high performance is still maintained. This demonstrates the benefits of combining a lower quality depth camera with our method for redistributing the planes. Unfortunately, the limited range of the Kinect limits its usefulness to only small scale scenes.

## 8.4 Results

Our immersive collaboration environment has been demonstrated as a fully functional prototype in real-world scenarios at several local conventions [Dumont et al., 2010, 2011]. An impression is given in Figure 8.13, which shows two networked immersive environments at the ServiceWave 2010 convention in Ghent, Belgium. User feedback from convention visitors was very positive overall. Many confirmed that eye contact was convincing and that the panoramic projection screen provided a true sense of immersion in the environment of the remote user. Furthemore, most visitors were able to intuitively operate the multi-touch surface, without any assistance whatsoever. The combination of all components into one immersive environment was judged to really facilitate collaboration at a distance and to give the sensation of a true office of the future.

At the time of development, the implementation of one immersive collaboration environment required at least two computers. One computer took care of the acquisition and rendering, while the other one performed the image stitching and deformation for the immersive projection. Both computers had an Intel Core 2 Quad CPU and an NVIDIA GTX 280 graphics board with 1 GB GDDR3 memory. The cameras were a mix of bus-synchronized Point Grey Grasshopper and Flea cameras with FireWire connections. The panoramic screen was projected on by three Optoma TX1080 DLP projectors. The multi-touch surface has its own embedded processor for interpreting multi-touch gestures and displaying media. The audio



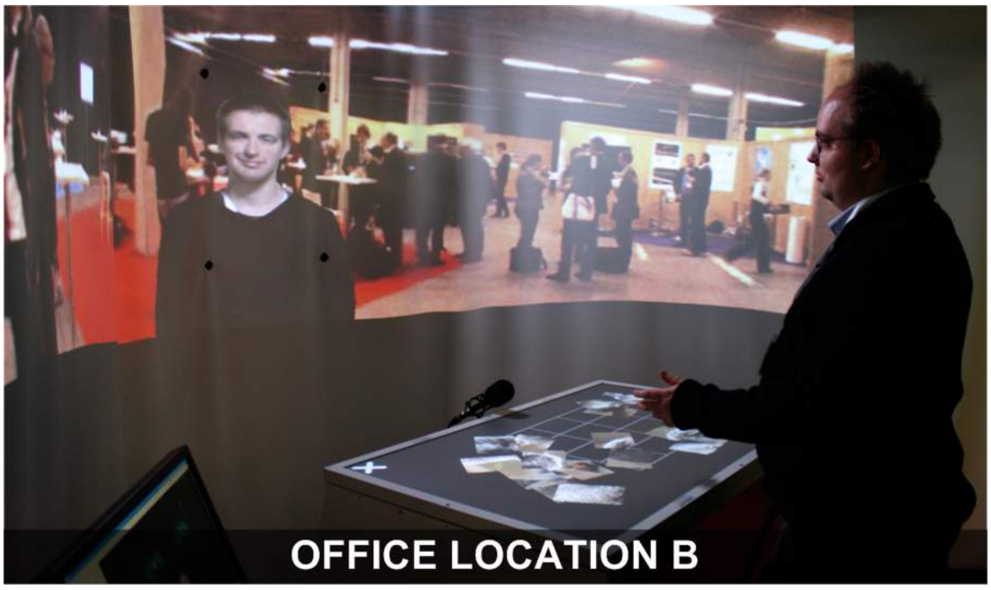


**Figure 8.13:** Our immersive environment enables collaboration between two locations.

processing and echo cancellation was also performed on a dedicated Mac mini. Everything combined, our system still achieved real-time performance at over 26 FPS.

## 8.5 Requirements Evaluation

This is our last solution to eye gaze correction, in this chapter expanded to full immersive collaboration at a distance. As always, the evaluation is carried out on the requirements defined in section 1.1, p. 3. The chart in Figure 8.14 is finally completed and now allows us to quickly get a feel of the strengths and weaknesses of each solution and how they relate to one another.

**Eye Contact** Eye gaze is corrected in the exact same way as in – and thus completely on par with – our plane sweeping prototype (chapter 7). However, we still score it one point lower, because the much larger user-to-screen distance causes the eyes to be less clearly visible. On the other hand, we could confirm that humans are very well adapted to pick up on eye contact at even the farthest of distances. **6/7 (very good)**

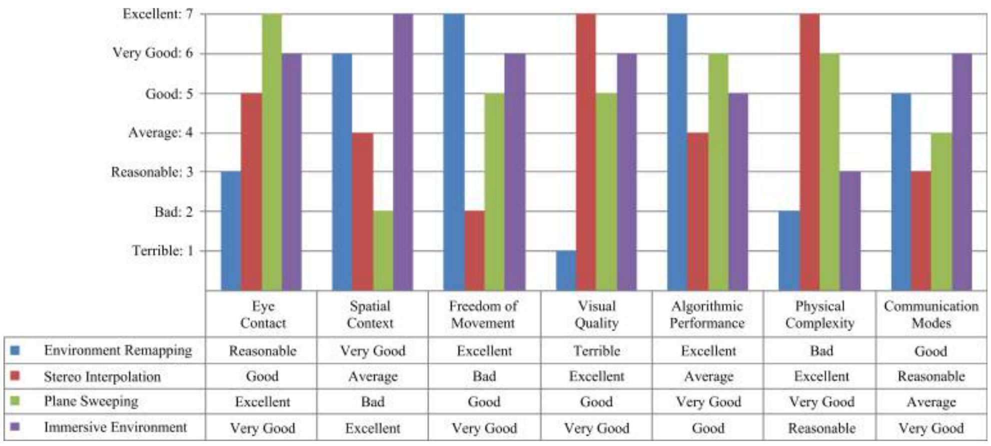
**Spatial Context** The missing background limitation of our plane sweeping prototype (chapter 7) is overcome by capturing the user's background with an omnidirectional camera and projecting it on a panoramic screen. This large panoramic screen engulfs the user's peripheral vision and provides him with the most extensive spatial context. Our other prototypes are no match for this. **7/7 (excellent)**

**Freedom of Movement** The user's freedom to move is only held back by the size of the panoramic screen and the area that is covered by the capturing cameras. **6/7 (very good)**

**Visual Quality** Our plane sweeping algorithm from chapter 7 is used to correct eye gaze. Consequently, there is virtually no change in visual quality, though the larger user-screen distance implies that occlusions are less severe and artifacts are less discernible. The addition of a panoramic background also helps to mask artifacts. **6/7 (very good)**

**Algorithmic Performance** Again, the same plane sweeping algorithm from chapter 7 is used, with little change in performance. The system is extended with panoramic background rendering, surface computing and audio processing. This increases the complexity, as all of these components must operate synchronously. Especially the panoramic rendering does require extra processing, but in our current implementation each of the components run on their own dedicated hardware and thus function independently of one another. **5/7 (good)**

**Physical Complexity** The combination of multi-touch surfaces and immersive video conferencing turns out to be especially suitable for the purpose of communication and collaboration at a distance. However, the whole setup takes a lot of space, has the potential to be very expensive, and consists of many separate modules that each need to be set up and maintained carefully. All properties that no doubt pose a challenge for



| | Eye Contact | Spatial Context | Freedom of Movement | Visual Quality | Algorithmic Performance | Physical Complexity | Communication Modes |
|---|---|---|---|---|---|---|---|
| 🟦 Environment Remapping | Reasonable | Very Good | Excellent | Terrible | Excellent | Bad | Good |
| 🟥 Stereo Interpolation | Good | Average | Bad | Excellent | Average | Excellent | Reasonable |
| 🟩 Plane Sweeping | Excellent | Bad | Good | Good | Very Good | Very Good | Average |
| 🟪 Immersive Environment | Very Good | Excellent | Very Good | Very Good | Good | Reasonable | Very Good |

**Figure 8.14:** Requirements evaluation of our immersive collaboration environment. The scores correspond to the scale defined in section 1.1, p. 3. All other solutions have been evaluated at the end of their respective chapters. The completed chart now allows us to compare them all at a glance.

the average user, yet are more than conceivable in a professional office environment. **3/7 (reasonable)**

**Communication Modes** Developed with a focus on one-to-one communication, but the large working space could support many-to-many communication with little problem. **6/7 (very good)**

## 8.6 Conclusion

We took our plane sweeping prototype for one-to-one eye gaze corrected video conferencing from chapter 7 and extended it to an immersive environment for collaboration at a distance. We implemented the fundamental criteria of *The Office Of The Future* as originally proposed by Raskar et al. [1998] and went beyond by adding multi-touch surface computing and advanced audio processing. Many of the limitations of the one-to-one prototype of the previous chapter have thereby been tackled. Mainly, our environment offers the users very detailed spatial context and more freedom of movement. Everything combined, we still easily achieve real-time performance.

We proposed an adaptation of the plane sweeping algorithm to more efficiently interpolate a scene that contains multiple dominant depths, e.g. when multiple users are present in our immersive environment. When the content of the scene is not distributed evenly, the plane sweeping algorithm may be searching for matches in depth ranges where no objects are actually present. Doing so not only reduces computational efficiency, but also increases the likelihood of mismatches and thus noise. Our method employs the cumulative histogram of the depth map reconstructed in the previous temporal frame (or actively captured by a lower resolution depth camera) to determine a more suitable non-uniform plane distribution that is denser in regions with objects and sparser in regions without objects. We tested our method on various input sequences with multiple users in view. We succeeded to reconstruct high quality depth maps while only requiring a low number of planes.

The combination of multi-touch surfaces and immersive video conferencing is remarkably well-suited for the purpose of collaboration at a distance. The networked multi-touch surfaces allow multiple users to interact with the same application simultaneously, while directly manipulating their own private physical interface. Unlike with co-located collaboration, the users don't have to stand in front of the same table or screen and thereby risk getting in each other's way, while at the same time the immersive environment brings a very convincing sense of telepresence to the – pun intended – proverbial table.

### 8.6.1 Future Work

We developed our immersive environment with support for one-to-one communication and collaboration in mind. The system's spatial (large panoramic screen, lots of space to move around) and algorithmic (many virtual viewpoints can be reconstructed) nature has big potential to seamlessly support multiple users together, but a couple of main issues will need to be addressed.

Users should be discerned and tracked, so that each can be offered their own viewpoint based on the position and orientation of their eyes. The tracker developed by Maesen et al. [2013] looks very applicable. It determines the six degrees of freedom of the head pose by passively tracking (infrared) LED strips mounted to the ceiling. It is independent of the size of the working area and puts no restriction on the number of users to track [Maesen, 2016].

Most challenging is showing a perspectively correct image to each user separately, from his own point of view, on the same panoramic screen. A good place to start looking for solutions is the work of Nashel [2010], who trades off stereoscopic viewing in autostereoscopic parallax displays to provide unique monoscopic images to multiple viewers in different positions.

# Chapter 9

# Sociability Study

## Contents

**Figure 9.1:** This chapter performs a sociability study on the users of eye gaze corrected video conferencing in the overview in Figure 1.1, p. 2.

Do users really need eye contact to communicate? How can direct eye contact enhance social interaction? Does eye contact play an important role in providing a feeling of telepresence? What other factors contribute to telepresence?

Preece [2001] states that sociability is concerned with *developing software, policies and practices to support social interaction online*. Although sociability is closely related to usability, it differs from it in the sense that usability is mainly geared at how users interact with certain technologies, whereas sociability is concerned with how users interact with each other using that technology. In other words, usability focuses on the interaction across the human computer interface, while sociability is concerned with interaction between humans supported by technology.

We borrow from the studies carried out in collaboration with CUO and MICT/SMIT on the sociability of several video conferencing technologies developed in the context of the IBBT ISBO VIN project [2005–2008] [Mechant et al., 2008; van Nimwegen, 2008]. CUO organized hands-on sessions and presented the participants with questionnaires to fill out afterward, while MICT/SMIT concurrently conducted a focus group conversation. An overview of both procedures is given in section 9.2 and section 9.3 respectively, but not before the evaluated technologies are first presented in section 9.1. Finally, we summarize relevant results of both studies in section 9.4 and we conclude in section 9.5.

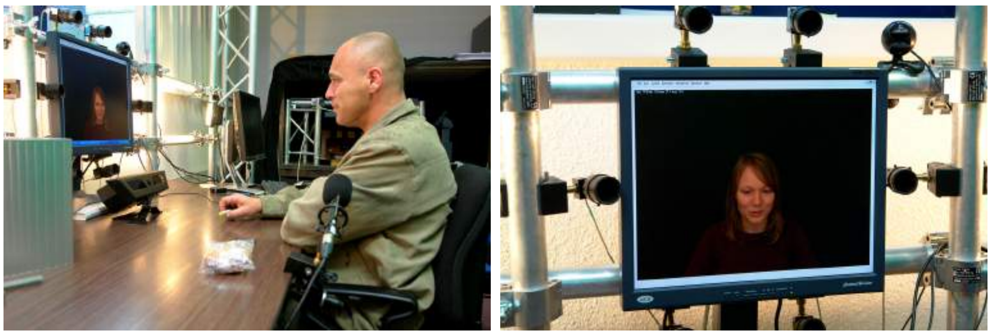
## 9.1 Evaluated Technologies

Mainly because of their readily availability as functioning demonstrators at the time, it was decided to focus on the two technologies presented here. Notice the interesting juxtaposition between our image-based solution in section 9.1.1 on the one hand and ETRO's model-based solution in section 9.1.2 on the other.

### 9.1.1 Our Face-To-Face Prototype

This is our end-to-end prototype for close-up one-to-one eye gaze corrected video conferencing as we developed it in chapter 7. We acquire live images from six cameras placed closely around the screen and use plane sweeping to reconstruct the eye gaze corrected image of a virtual camera that is positioned by an eye tracker [Dumont et al., 2008, 2009b]. Figure 9.2 shows our prototype as it existed at the time of evaluation.

While it is true that continuous eye contact is provided, this comes at the expense of realism. The algorithm reconstructs an image that is in a sense a blend of all the surrounding cameras and could thereby be perceived as less sharp – less *crisp* – than the image that would be captured by a single camera only. Visual artifacts do occur, and more than that, no context information on the user's environment is available as a result of background subtraction (recall Figure 7.15, p. 151), though this is more a limitation of the prototype than the system concept as a whole.



(a) Unnamed test participant. (b) Conversation leader Sophie.

**Figure 9.2:** (a) An unnamed test participant is conversing on site using our face-to-face prototype (UC2) with corrected eye gaze. (b) A close-up shows the conversation leader Sophie and the single webcam (UC1) in the top right corner.

### 9.1.2 ETRO's Facial Communication

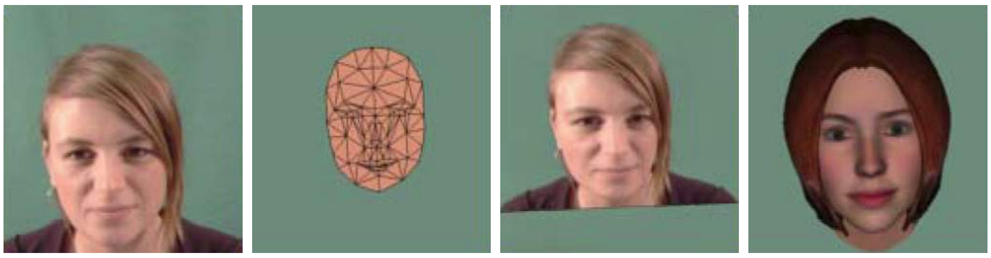
ETRO's solution employs a 3D vector-based representation of 2D video images. From the original 2D video in Figure 9.3(a), they analyze a sequence and reconstruct a 3D model of the user by identifying the 28 characteristic points marked in Figure 9.3(b). These 28 points are then tracked by a camera and their displacements are applied to a real-life resembling model of the user, resulting in the enhanced reality video stream in Figure 9.3(c). Alternatively, they animate an artificial avatar to represent the user, as shown in Figure 9.3(d).

This technology is particularly interesting in mobile contexts or other situations where only low bandwidth is available, since the vector-based representation allows the complexity of the model to be adjusted according to the available bandwidth. However, the face one communicates with is no longer an exact photorealistic representation. It is a reconstructed 3D model and not an image-based rendered assembly of multiple frames.

## 9.2 Hands-On Sessions

After hands-on experience with the technology prototypes, CUO asked the participants to fill out questionnaires [van Nimwegen, 2008].

We begin with deducing four use cases in section 9.2.1. They are evaluated according to the experimental design described in section 9.2.2. Through each use case, two users (the experimenter and a test participant) hold a conversation that is designed to be as natural as possible, as described in section 9.2.3. Finally, how and what data is measured is explained in section 9.2.4.



(a) Original video. (b) 3D modeling with motion estimation. (c) Enhanced reality. (d) Animated avatar.

**Figure 9.3:** Steps to the enhanced reality (UC3) and animated avatar (UC4) facial communication prototypes, developed by ETRO.

### 9.2.1 Four Use Cases

From the video conferencing technologies presented in section 9.1 and in combination with an extra control use case, four use cases emerge to evaluate and compare:

**(UC1)** **Single Webcam**: This is the control use case that serves as a baseline reference. A conventional video conferencing situation is set up, where two participants are communicating with each other, each behind their own terminal with a single webcam that consequently cannot provide eye contact. No special processing (e.g. compression, overlay rendering or avatar representations) are applied. In essence this is the video chat situation that people are most familiar with at home or at work, as introduced in this dissertation's problem statement in Figure 1.2, p. 3. The webcam can be noticed in the top right corner in Figure 9.2(b).

**(UC2)** **Our Face-To-Face Prototype**: Our prototype provides close-up face-to-face communication for two users. The six cameras around the screen and the applied rendering allows for continuous and real-time eye contact. This is our video conferencing prototype exactly as developed in chapter 7. Figure 9.2(a) shows an unnamed participant testing this setup.

**(UC3)** **ETRO's Enhanced Reality**: This is ETRO's solution from Figure 9.3(c), where 28 characteristic points are detected and tracked by a camera and their displacements are applied to a reconstructed 3D model of the user's face.

**(UC4)** **ETRO's Animated Avatar**: Similar to (UC3), this is ETRO's solution from Figure 9.3(d), but instead of using a model of the user's face, the animations are applied to an avatar representation.

At the moment of evaluation, ETRO's facial communication technology, both with real face model (UC3) and animated avatar (UC4), was not available for use in real-time, because of the intensive processing involved. Nevertheless, by processing prerecorded audiovisual data, the technology became feasible for use in the experiments in such a manner that users were hardly, if at all, aware that the act was not happening live.

### 9.2.2 Experimental Design

To the extent possible, the four use cases should be evaluated on equal dimensions and in a social setting. A within subjects experimental design was opted for, meaning that all test participants will experience all four setups.

A professional (henceforth referred to as Sophie) was contracted to act as the conversation partner in all user sessions and in all four use cases. The risk of influencing the results by having another conversation partner each time is thereby minimized.

As test subjects, 43 persons were recruited via a recruitment agency. Most of them lived in Hasselt or somewhere in the province of Flemish Limburg; 24 were male, 19 were female, with ages ranging between 19 and 69 years. All were financially compensated for their participation.

For each use case, two terminals were set up in two separate rooms. Sophie constantly manned one terminal in one room, while each test subject was one by one invited into the other room. Sophie then proceeded to lead a conversation with the test subject, so that the latter could experience and evaluate the use case in question.

It should be noted that due to practical circumstances the ETRO material was delivered later than planned, when the lab was already set up and arranged. As a consequence, all 43 subjects could use the single webcam (UC1) and our face-to-face (UC2) setups, but only 13 could test ETRO's enhanced reality (UC3) and 37 could test ETRO's animated avatar (UC4) technologies. This is corrected for when scoring the questionnaire results.

### 9.2.3 The Conversations

For each of the four use cases from section 9.2.1, a conversation between Sophie and a test participant is conducted via the technology of that particular use case. The conversation itself in turn consists of four parts during each use case, with some parts differing from use case to use case. From Sophie's side, the content of the conversation is practically identical time after time, whereas the answers given and stories told by the participants obviously vary. All conversations were carefully prepared and practiced to guarantee that they would go smoothly, without answers from participants swinging off, and to ensure they were suitable for the general research goal. The four parts of the conversation are as follows.

#### 9.2.3.1 Listening

Sophie introduces herself and tells an everyday story that the participant listens to. These stories differ in each use case and are meant to be easy to relate to by the listener. They are:

(UC1) Single Webcam: How she got there this morning (get up, take bus, walk, etc.).

(UC2) Our Face-To-Face Prototype: What her day was like yesterday (activities, work, dinner, etc.).

(UC3) ETRO's Enhanced Reality: Describes her house, where she lives.

(UC4) ETRO's Animated Avatar: Her last vacation / trip.

#### 9.2.3.2 Speaking

After her particular story above has been told, Sophie now asks the participant to tell her their story on the same subject, i.e. how they got there, their day yesterday, their house or their vacation. Again, the conversation topics were chosen so that they could easily be answered by all participants.

#### 9.2.3.3 Short Conversation

The previous two parts of the conversation (respectively listening and speaking) were unidirectional, that is to say an uninterrupted flow from Sophie to the participant or vice versa. In contrast, the short conversation presented here goes back and forth and is formulated by Sophie in such a way that it is easy to provoke interaction from the participant. It deals with:

(UC1) Single Webcam: About the pros and cons of toll roads in Belgium. Sophie asks simple questions to inquire about the participant's opinion. What about other countries where this happens? Do you like the idea? Would you consider taking public transport instead?

(UC2) Our Face-To-Face Prototype: About a Belgian TV show that everybody knows: F.C. De Kampioenen. Why is this show so popular? What else do you like to watch? Do you know this other new series?

(UC3) ETRO's Enhanced Reality: About a (non-existing) newspaper article about people escaping from prison. Did you read it? Do you know how prisoners most commonly escape? What measures can you think of to prevent escape?

(UC4) ETRO's Animated Avatar: About plane tickets. Where to find a cheap ticket to New York? Is it advantageous to book the flight and accommodation separately? Where would you go if you would be offered a free flight to anywhere right now?

This short conversation could also be performed with ETRO's prerecorded audiovisual fragments (see section 9.2.1), because questions were posed in such a manner that answers were always short and to the point, and Sophie's reply – although prerecorded instead of live – fitted in as if it was real and natural.

#### 9.2.3.4 Long Conversation

This conversation is most realistic, in the sense that the course of the conversation is truly free. For this reason, this part is only possible in the live setups, namely the single webcam (UC1) and our face-to-face (UC2) use cases. It was decided to casually discuss railway strikes and global warming, both subjects on which people usually have an opinion or are at least familiar with.

### 9.2.4 Measures

Data was collected and measured by the following means.

**Questionnaires** After running through each of the four use cases, the participant fills out a questionnaire on which they have to score several aspects of the tested system on several dimensions. The questionnaire is included in Appendix A, Table A.1, p. 211.

**Informal Remarks** Although the conversations from section 9.2.3 are fairly structured, they are still allowed to flow naturally to the extent possible. As such, the conversations are sometimes steered so that certain relevant issues are informally mentioned by the participants.

**Eye Tracking** A Tobii Technology eye tracker is placed below the screen. It measures where, for how long and how often the participant looks at the screen. It would have been interesting to compare this across the different use cases, but in practice the eye tracker turned out to be less accurate than hoped for. Very few differences could be determined between the use cases, which rendered it rather difficult to interpret and infer any meaningful results.

## 9.3 Focus Group

A focus group on video conferencing was conducted by MICT/SMIT [Mechant et al., 2008]. Focus groups are a convenient way to collect qualitative and focused research data. In essence they exist of controlled group interviews, whereby a moderator leads the interview that allows a small group to discuss the subjects that are brought forward. They are more flexible, open and less standardized than surveys and thereby allow for easy exploration, insight into context and depth of the subject, and creation of an interpretative framework [Morgan and Krueger, 1997].

### 9.3.1 Discussion Topics

The focus group was guided by a moderator who kept to a semi-structured topic list:

- Exploratory Part: The participants were asked about their vision and opinions regarding communication via video conferencing. It was explored how the participants have integrated video conferencing into their home or professional environment, which tools they use, the number and duration of video conferencing sessions and associated gratifications.
- In-Depth Part: Details that were discussed during the exploratory part were delved into. Questions arose such as: What is the role of video conferencing in communities? Does the visibility benefit the group dynamic? Is video conferencing mainly used (in private) between two people or in (a public) group? Why or why not?
- Hands-On Session: A short break during which the participants could experience our face-to-face prototype (UC2) firsthand.
- Analysis Part: After the hands-on session, the participants were asked to systematically identify values, bottlenecks, issues and wishes related to our prototype, using post-its to make the session (inter-)active. Values refer to the added value that participants see in communicating using video conferencing instead of other communication channels. Bottlenecks refer to things that may hinder or impede the communication. Issues are points that the participants consider important for efficient online audiovisual communication. Wishes express what the user would ultimately like if he would have unlimited resources.

Two extra observers were present, the whole conversation was recorded on a digital audio recorder and finally transcribed.

### 9.3.2 Group Composition

With the help of an external recruitment agency, a purposive and convenient sample of participants was composed. In a purposive sampling, the participants are chosen in function of the goal of the study. In this case, it means only people who have been active online and have used video conferencing in one way or another were selected. Convenient sampling entails that the sample is only to a limited extent attempted to be made an accurate representation of a larger group or population.

Unfortunately, due to significant problems experienced by the recruitment agency, only four of the eight promised recruits showed up to participate in the focus group: three men and one woman, born between 1948 and 1988. All participants received financial compensation for their time.

Due to the limited size of the sample, it is difficult to generalize the results from the focus group to the entire population. The methodological and practical requirements of a focus group in the strict sense of the word were arguably not met. However, it was decided to go ahead with the session anyway. Even though the results should therefore be considered to be purely exploratory, some interesting and striking findings emerged nevertheless.

## 9.4 Results

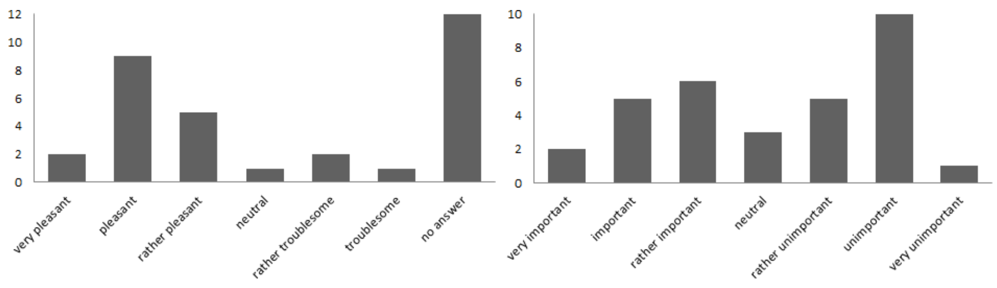
For a detailed analysis of the answers given on the questionnaires after the hands-on sessions, refer to Table A.1, p. 211, and Table A.2, p. 213, in Appendix A. We continue to make some key observations, integrating relevant results of both the hands-on sessions (section 9.2) and focus group (section 9.3).

Looking at the raw numbers, and perhaps somewhat against our hopes and expectations, we must note that the single webcam (UC1) system appears to emerge as the participants' preferred video conferencing solution. Our face-to-face (UC2) and ETRO's enhanced reality (UC3) systems, although both not without their quality flaws, were often judged similar, but also judged slightly lower than the single webcam (UC1) system. However, when GLM (generalized linear model) shows the scores to differ significantly, our face-to-face prototype (UC2) consistently scores a close second, with ETRO's enhanced reality (UC3) system coming in third and their animated avatar (UC4) solution closing the ranks. Even more so, post-hoc comparison often reveals our face-to-face prototype (UC2) to not differ significantly from the single webcam (UC1) setup (cf. Questions 1 and 4–8).

The majority of participants do confirm the provision of eye contact to be pleasant, when presented with the question in Figure 9.4(a). Also during the focus group discussions, the participants judged eye contact to render the communication more intense, personal and direct. In spite of this, eye contact was not overwhelmingly judged to be absolutely indispensable, but rather more of a technological gimmick. One reason that was heard – simple yet easily overlooked – is that most people cannot type blindly and are therefore used to losing eye contact anyway when looking at the keyboard (if audio for voice communication is not used or not available). We did observe that the eye tracker in our face-to-face prototype (UC2) at least made the restoration of eye contact feel natural after looking away.

An interesting point to look at is the influence of image quality on communication quality. Although all view interpolation algorithms, including our own, strive to synthesize an image that is as free of artifacts as possible, one would be hard-pressed to deny that the absolute perfect algorithm does not (yet?) exist. To this end, Questions 7–10 (pleasantness and effectiveness) combined with Question 11 (image quality) indicate that image quality is a deciding factor in the appraisal of a video conferencing system. In contrast, our test subjects indicate in Figure 9.4(b) that having the image appear at a true-to-life size is less important. Others stated they would prefer a high-definition image over one that is free of artifacts. Clearly, room is left for research on having eye contact at the expense of image quality in social contexts. In one instance, Yang et al. [2006c] do report that low image resolution makes it more difficult to seek eye contact in their immersive environment for collaborative dancing.

The participants often expressed their concern about some technical characteristics of the demonstrators themselves. It was heard that the physical size of the demonstrator (e.g. many cameras, heavy workstation pc) makes it impossible to carry it around to show the environment, as people often do with a smaller laptop or phone. A valid point was also made in that



(a) Is having continuous eye contact pleasant or rather troublesome?

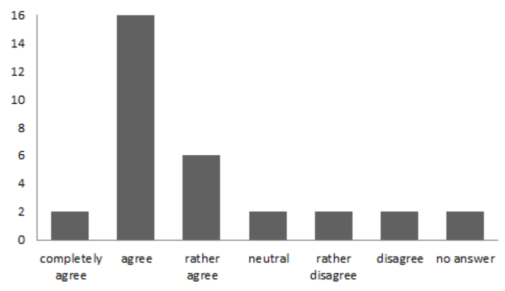
(b) How important is it that the rendered image of the conversation partner is at true size?

**Figure 9.4:** Participants' remarks on our face-to-face prototype (UC2).

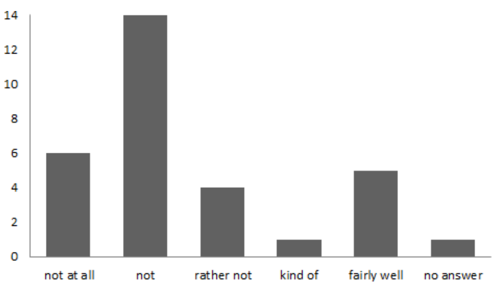
eye contact is often broken purposefully to direct one's attention to what's happening in the environment and thus continuous – unbroken – eye contact might not be an unconditional requirement. On a similar note, participants expressed a desire for more environmental context in order to gain a more natural feeling of presence and receive more social cues. We must keep in mind, however, that the participants often struggled to imagine the demonstrators as a first step toward a finished marketable product, free of the technical complexities that can plague a prototype. Nevertheless, these issues prove relevant to our research, as they essentially describe the spatial context, freedom of movement and physical complexity requirements (defined in section 1.1, p. 3) that we so often returned to throughout this dissertation.

The focus groups also brought forward that audio contributes quite significantly to the overall experience. Moreover, regarding performance, it was stated multiple times that delays in audio are much more detrimental to efficient communication than delays in video.

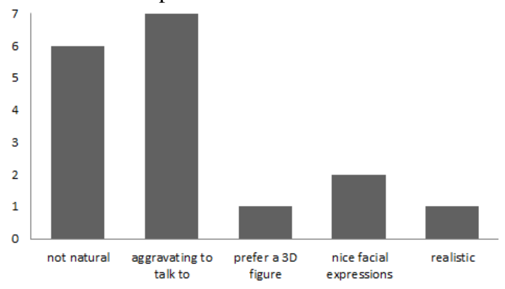
Throughout this dissertation, we have intentionally chosen to correct eye gaze using image-based rendering techniques (environment remapping, rectified stereo and plane sweeping). We assumed that they would produce a more true-to-life image than methods that perform model-based reconstruction, thereby especially avoiding the uncanny valley effect that artificial reconstructions of human faces so often suffer from [Mori et al., 2012]. Our choice now appears to be warranted by our face-to-face prototype (UC2) that has consistently scored higher than ETRO's enhanced reality (UC3) and animated avatar (UC4) solutions. Although all measured differences are significant, the difference is not terribly outspoken for ETRO's enhanced reality (UC3) system, which also processes live captured imagery and textures it on a 3D reconstructed model. However, for ETRO's animated avatar system, which does not output any live captured imagery at all, the difference is very apparent. In particular take a look at Questions 1 (engagement), 2 (physical presence), 5 (lifelikeness) and 13 (intimacy), indicating that a lifelike representation of the conversation partner contributes significantly to experiencing the video conferencing technology as natural and engaging. This is definitely



(a) The avatar representation impacts the conversation.



(b) Does the avatar representation of Sophie resemble the real person?



(c) Other various remarks on ETRO's animated avatar system.

**Figure 9.5:** Participants' remarks on ETRO's animated avatar (UC4) system.

also reflected in the informal remarks in Figure 9.5, with the majority of the participants agreeing that the avatar representation impacts the conversation and that it does not mimic the real-life appearance of Sophie well. Most participants also find the avatar unnatural and aggravating to talk to.

## 9.5 Conclusion

In collaboration with CUO and MICT/SMIT, a concise study was performed on the sociability of several video conferencing technologies. The study consisted of hands-on sessions and a focus group discussion. We summarized their modus operandi and collected the results that are most relevant to the eye gaze correction problem treated in this dissertation. For a more extensive discussion and analysis, including on the contribution of 3D audio to immersive video conferencing, please consult the originating reports [Mechant et al., 2008; van Nimwegen, 2008].

What was brought forward most by the participants is that, although eye contact is definitely appreciated, it is not the only property that is looked for in a video conferencing system. It was suggested that factors such as image quality, spatial context, freedom of movement and physical complexity, which are in fact the requirements that already emerged naturally from our prototypes throughout this dissertation (defined in section 1.1, p. 3), also contribute to establishing a social telepresence.

Our prototype using plane sweeping (as developed in chapter 7) was consistently scored higher than solutions employing a model-based reconstruction or an avatar-based representation of the conversation partner. We pose that an image-based approach produces a more lifelike representation of the conversation partner, avoiding the uncanny valley and contributing to a more natural and engaging video conferencing experience. This warrants our choice for image-based rendering algorithms (environment remapping, rectified stereo, plane sweeping) throughout this dissertation.

We end on a final thought about not observing a too outspoken difference in the appraisal of having eye contact. It might be that people tend to simply accept technology as it is, meaning that a real face-to-face meeting is experienced as a wholly other thing and that we are happy with video chat just as it is. After all, this is technology that until not too long ago belonged to the realm of science fiction and perhaps, for example, Skype video in its current state makes us happy enough already.

### 9.5.1 Future Work

The evaluated prototypes were challenging in a technical respect. It would be interesting to work with technologies that have matured to get rid of imperfections outweighing for example the presence of eye contact. Furthermore, the systems might have been judged more differently in situations where people that actually know each other use them, and in situations such as distance education or business meetings/negotiations where eye contact and trust is more fundamental. Finally, to arrivate at the ideal video conferencing solution, we require more insight into the concept of presence, what it means to experience a virtual telepresence and exactly what factors enable this experience. Is this eye contact, context information, or are other social elements decisive?

# Chapter 10

# Conclusion and Future Work

## Contents

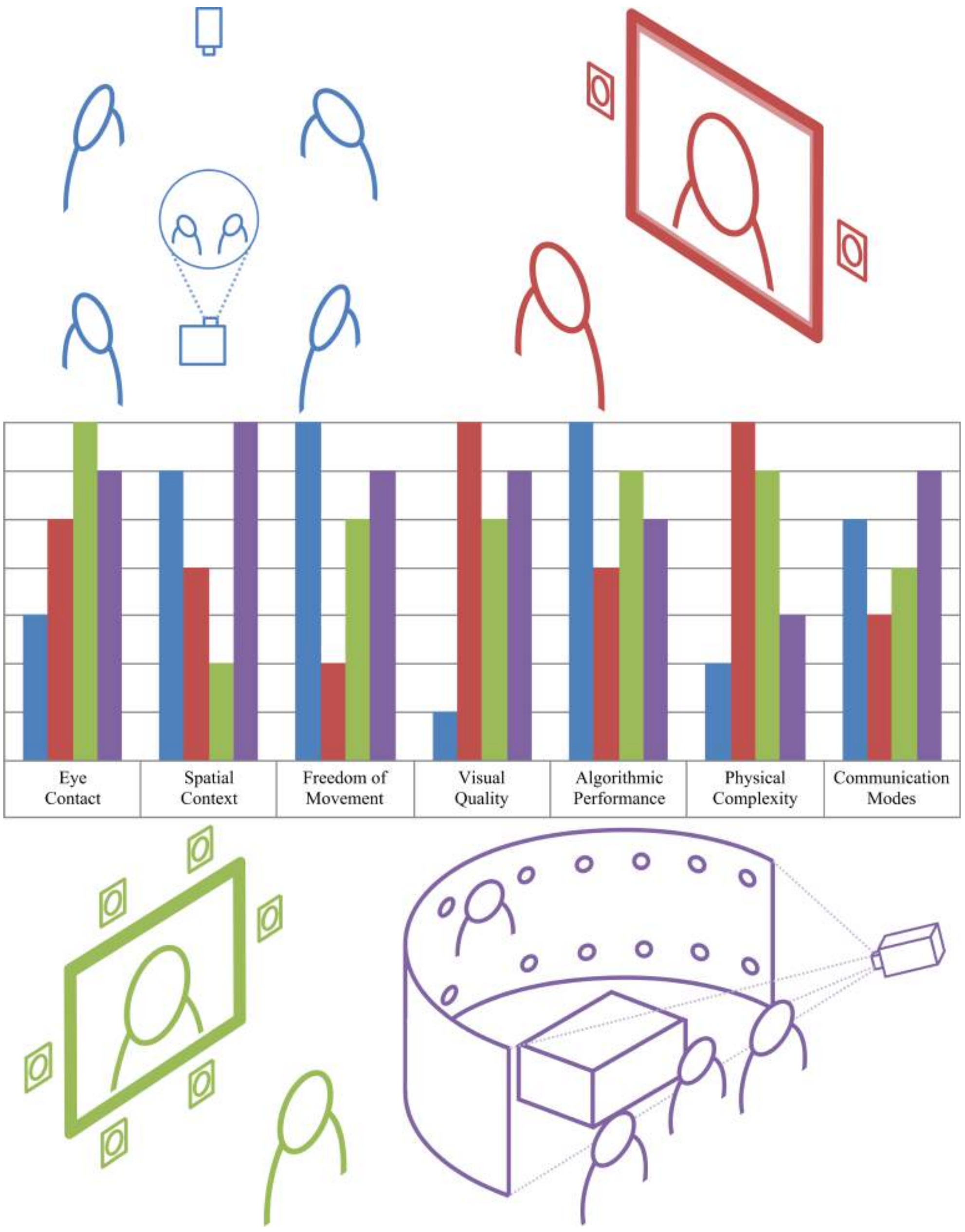


**Figure 10.1:** We developed real-time image-based rendering algorithms to correct eye gaze in video conferencing. We implemented our solutions in four different prototypes (cf. Figure 1.1, p. 2) and evaluated them on seven requirements (cf. section 1.1, p. 3). This chapter concludes by discussing and comparing all prototypes and their underlying image-based rendering algorithms one final time.

In this dissertation, we identified the central problem of missing eye contact in video conferencing. Eye contact is lost because the user cannot simultaneously look at the screen and into the camera that is commonly located in the vicinity of the screen. Furthermore, due to the narrow field of view of a single camera, only very limited context information on the user's environment is available. This lack of eye contact and context information severely impedes communication and collaboration.

## 10.1 An Image-Based Approach

We set out to correct eye gaze by purposefully taking an image-based approach, meaning that we synthesized a novel eye gaze corrected image from images that are (live) captured by cameras.

In chapter 2, we gave an overview of image-based rendering algorithms and situated our work in a spectrum with a (accuracy of) geometry versus (number of) images trade-off. In the same chapter, we also explained how to exploit the GPU for general-purpose computations (GPGPU) and thereby take advantage of its massive parallel processing capabilities. By designing and implementing all our view synthesis algorithms for and on the GPU, their real-time performance and future-proof scalability is guaranteed.

In a concise sociability study in chapter 9, we compared (in cooperation with CUO and MICT/SMIT) one of our own image-based solutions with technologies that rely on geometric modeling and on avatar representations (developed by ETRO). The study showed that users prefer the inherent realism offered by image-based approaches over the more artificial looking geometric and avatar reconstructions. This warrants our choice for image-based rendering, as it avoids the *uncanny valley* effect that artificial reconstructions of human faces so often suffer from [Mori et al., 2012].

## 10.2 Four Prototypes and Seven Requirements

We developed four different video conferencing prototypes, with each prototype taking its own specific image-based rendering approach to correcting eye gaze.

As each image-based rendering algorithm requires a different physical configuration of cameras, we were able to arrange our prototypes according to increasing complexity of their camera setup. This arrangement – first depicted in Figure 1.1, p. 2 – formed the leading thread throughout this dissertation. The conceptual drawings of the four prototypes are repeated in the four corners of Figure 10.1.

From experience with our prototypes, the same seven requirements that any ideal solution should fulfill kept naturally emerging. They have been defined in the problem statement in section 1.1, p. 3, and are: eye contact (and the related gaze awareness), spatial context, freedom of movement, visual quality, algorithmic performance, physical complexity, and communication modes. The participants of the sociability study in chapter 9 also pointed toward the importance of these requirements, although further research is desired to gain insight into the concept of presence, what it means to experience a virtual telepresence and exactly what factors enable this experience.

We evaluated all prototypes on the requirements at the end of their dedicated chapters. The final score chart is shown in Figure 10.1(middle) and acts as a summary of the whole dissertation. The colored bars correspond to a score with a 7-point scale that we informally defined in section 1.1, p. 3: (1) terrible, (2) bad, (3) reasonable, (4) average, (5) good, (6) very good, (7) excellent.

Guided by the identified requirements, we will now discuss and compare all prototypes and their underlying image-based rendering algorithms one final time. However, note that lots of requirements are interdependent and many trade-offs exist. Keep in mind, also, that their evaluation is an informal one and we should be careful not to compare apples and oranges too much. Nevertheless, they allow us to quickly get an idea of each prototype's strengths and weaknesses in a consistent manner. Moreover, the seven requirements provide a reference framework around the experience gained in this dissertation on which to design, develop and evaluate any future solution to eye gaze corrected video conferencing.

### 10.2.1 Environment Remapping (One Camera)

In chapter 4, our first prototype implemented a somewhat unconventional solution that requires only a single camera and that is based on the concept of environment mapping. We captured omnidirectional video (in other words, the environment) by filming a spherical mirror (the northern hemisphere) and combined this – after a remap of the captured image – with projection on an identically-shaped spherical screen (the southern hemisphere). Both capture and display hemispheres are pasted together into a single sphere, forming the single communication device depicted in Figure 10.1(top-left). The unconventional novelty lies in the observation that we do not perform image interpolation in the traditional sense, but rather compose an eye gaze corrected image by remapping the captured environment pixel-to-pixel. This image transformation is completely independent of the scene structure and does not require the recovery of the depth of scene. Instead, it relies on the mathematical equations that map the captured input to the projected output, both interpreted as parallel rays of light. Because unfolding the environmental reflection captured on a (small, relative to the environment) specular sphere yields omnidirectional imagery with a projection center located at the center of that sphere, the user looks directly into the camera when looking at the center of the sphere and eye contact is inherently guaranteed.

The pixel-to-pixel image transformation has to be computed only once, resulting in a very fast and lightweight implementation (algorithmic performance: 7/7, excellent). Image quality, however, suffers severely due to the use of off-the-shelf hardware components (imperfect store surveillance mirrors) and limitations of the mathematical model (visual quality:

1/7, terrible). Nevertheless, when pixel resolution was high enough to see the eyes of the remote user, we have witnessed eye gaze to be accurate (eye contact: 3/7, reasonable). A more rigorously built system with quality mirrors, high-definition cameras and a precise calibration should be able to improve the visual quality and confirm the provision of eye contact empirically.

The omnidirectional nature of the device allows many users to communicate simultaneously (communication modes: 5/7, good), while the spherical screen unveils their environment to the full extent (spatial context: 6/7, very good). The users also possess an unprecedented freedom of movement in the entire space around the device (freedom of movement: 7/7, excellent). However, the physical size of the whole setup is not the most convenient, requiring a rather large camera/projector to sphere distance (physical complexity: 2/7, bad).

The image remapping equations are governed by an affine camera model. Using perspective cameras might complicate calculations to some extent, but it should be able to alleviate the vanishing point artifacts to some degree. Any missing image information at the vanishing point coordinates can then be interpolated. We also assumed a known external calibration of the camera-sphere-projector configuration. Achieving this alignment proved to be an elaborate manual process. An automatic calibration would decrease the setup time and at the same time would remove image distortions originating from misalignment. The work of Svoboda [2000], Pajdla et al. [2001] and Francken et al. [2007] is a good place to start.

### 10.2.2 Stereo Interpolation (Two Cameras)

We developed our second prototype in chapter 6, this time focusing on close-up one-to-one communication. We set up a camera to the left and to the right of a conventional computer screen, as depicted in Figure 10.1(top-right), captured the user that was seated in the horizontal middle and employed rectified stereo interpolation to reconstruct the eye gaze corrected viewpoint. We followed the depth-image-based rendering pipeline as formalized by Scharstein [1996] and Rogmans et al. [2009c], which essentially consists of a disparity estimation and view synthesis stage.

The view synthesis is extremely lightweight, but relies heavily on accurate disparity estimation for the input images to be warped correctly to the intermediate viewpoint. In chapter 5, we therefore first developed a novel edge-sensitive local stereo matching algorithm with iterative disparity refinement. We also introduced the idea of hierarchically limiting the disparity search range to increase the matching quality while at the same time decreasing algorithmic complexity. We will discuss both in more detail in section 10.4 and section 10.5 respectively.

The final eye gaze corrected image is virtually indistinguishable from the image interpolated by the (industry standard but not real-time) MPEG reference software. It is very sharp and contains few disturbing artifacts (visual quality: 7/7, excellent). A substantial amount of the user's environment is also interpolated, in spite of most information that is not visible in both cameras being lost. Nevertheless, it is difficult for stereo interpolation to offer more context than an ordinary webcam can (spatial context: 4/7, average).

The entire processing pipeline, in particular including our stereo matching algorithm, has been designed for efficient GPU implementation in CUDA. However, our stereo matching algorithm – although at the level of many other state-of-the-art stereo matching algorithms – still has an impact on the overall performance that is not to be underestimated. In fact, it causes our stereo interpolation prototype to be the least performant compared with our environment remapping and plane sweeping approaches (algorithmic performance: 4/7, average).

The user's freedom to move is restricted to the horizontal baseline between the left and right capturing cameras (freedom of movement: 2/7, bad). Although the eyes are clearly discernible, this causes eye contact to be difficult to maintain, as it depends on the user remaining stationary in the sweet spot of the reconstructed viewpoint (eye contact: 5/7, good). It also means that only one user can be active in front of the screen at the same time (communication modes: 3/7, reasonable). Of course, the potential to offer convincing parallax effects if the user does move (and the reconstructed viewpoint follows) is an advantage over an ordinary webcam that contributes to immersivity. Moreover, modern cameras are small and cheap, can easily be integrated into the bezel of any display and can be factory-calibrated to be in perfect rectified stereo (physical complexity: 7/7, excellent).

The biggest limitation that we encounter is that dense stereo matching is geared toward small-baseline camera configurations and therefore is not optimally suited for wider-baseline applications such as ours. In principle we wish to interpolate cameras that are spaced relatively far apart, i.e. to the left and right of a wider computer screen. Additionally, for our application there is always a screen-to-user distance trade-off, as the closer the user is to the screen, the larger the occlusions that arise will be. To alleviate this small-baseline limitation, more advanced occlusion handling algorithms could be looked at. Among them are (mostly global) methods based on segmentation [Bleyer and Gelautz, 2005; Wang and Zheng, 2008], graph cuts [Kang et al., 2001; Deng et al., 2007], belief propagation [Sun et al., 2005; Yang et al., 2009] and dynamic programming [Wang et al., 2006b].

In spite of all its limitations, we assess stereo interpolation to still be very suitable for other situations where eye gaze needs to be corrected. We imagine scenarios with a larger relative screen-to-user distance, e.g. video conferencing in a meeting room, or where the input cameras can be put closer together, e.g. integrated in the bezel of a smartphone or tablet. Applications outside of eye gaze correction are of course also conceivable, e.g. content generation to drive autostereoscopic 3D displays or scene interpretation for autonomous robotics.

### 10.2.3 Plane Sweeping (Multiple Cameras)

In chapter 7, we mounted multiple cameras around the screen, as depicted in Figure 10.1 (bottom-left), and turned to plane sweeping to overcome the limitations of rectified stereo interpolation. We specifically developed the prototype to efficiently support close-up one-to-one communication. However, the dense camera configuration could also support limited many-to-many communication, given a large enough (autostereoscopic) screen (communication modes: 4/7, average).

Plane sweeping can be regarded as a more generalized form of rectified stereo interpolation. It can handle cameras in a more general configuration with wider baselines. Instead of being restricted to the narrow horizontal stereo baseline, it allows us to synthesize high-quality images from any freely selectable viewpoint, without the need of image extrapolation. A concurrently running eye tracker positions the desired viewpoint and ensures that eye contact is maintained at all times (eye contact: 7/7, excellent).

Plane sweeping and rectified stereo are both image-based rendering methods that implicitly determine the depth of scene from their input images. For plane sweeping, however, the recovery of the scene depth is not essential, but rather a by-product of a rendering process that is based on color consistency. Even so, the whole processing pipeline exhibits a structure similar to the one of our stereo interpolation prototype. The view interpolation mainly consists of a depth reconstruction, refinement and recoloring stage. The refinement stage takes the reconstructed depth map and efficiently detects and removes outliers (i.e. artifacts) in the depth map using an efficient iterative spatial filter kernel. The recoloring stage subsequently repaints the refined depth map using one of two recoloring schemes. The N-camera recoloring scheme averages all input colors, thereby slightly blurring the result. The confident-camera recoloring scheme avoids occlusions by using the reconstructed depth map as a geometry proxy. It picks the best color based on angular distance to the input cameras, resulting in a far sharper image, but at the risk of making variations in illumination and photometric calibration stand out (visual quality: 5/7, good).

During the depth reconstruction, the overall algorithmic complexity is kept in check by a complexity control module that condenses the planes around the user's head and torso (discussed further in section 10.5). Combined with the eye tracker to position the virtual camera, this allows the user to move around much more freely in front of his screen, especially compared with our stereo interpolation prototype (freedom of movement: 5/7, good).

To overcome artifacts caused by occlusions that arise from the very small user-to-screen distance, we segment the input images into foreground and background and interpolate only the foreground (i.e. the face). This improves visual quality, but prevents us from reconstructing the user's environment (spatial context: 2/7, bad). However, we later corrected this in our immersive collaboration environment.

A number of design and implementation choices result in a very performant system (algorithmic performance: 6/7, very good). First, plane sweeping can match multiple cameras simultaneously by design. Rectifying the input images happens inherently during their projection on the planes. This significantly increases the reliability of pixel-to-pixel matches and diminishes the need for expensive aggregation of matching confidence scores. This is in stark contrast to our stereo matching algorithm, where the aggregation demands nearly all of the total processing time.

Furthermore, our implementation harnesses the computational resources of the graphics hardware to achieve comfortable real-time performance. This time, we configured the traditional graphics rendering pipeline through OpenGL and reprogrammed it for general-purpose computations by defining custom shaders for the vertex and fragment stages. Compared with stereo matching in CUDA, this approach lends itself better to the inherent structure and scattered memory access patterns of plane sweeping.

We further improved the end-to-end performance by maximizing arithmetic intensity and by introducing granular optimization schemes that map well to the polygon-based processing of the traditional graphics pipeline, without noticeable loss of visual quality. We also determined a fine-tuned set of parameters that are user-independent and that thus grant the system a general applicability. Lastly, network communication is brought to a minimum by sending the local user's eye coordinates to the remote user. This eliminates the need to send the captured images themselves over the network. Instead, they can be processed locally to interpolate the eye gaze corrected image from the correct viewpoint.

We designed and constructed a lightweight metal frame on which multiple (six, in our prototype) cameras can be mounted closely around the screen. As with our stereo interpolation prototype, integrating the cameras into the monitor bezel would introduce a fixed one-time calibration and thus remove the need for complicated offline calibration that requires user intervention. On the other hand, we would still have to integrate multiple instead of only two cameras. Being watched from nearby by multiple cameras might deter some privacy-sensitive users (physical complexity: 6/7, very good).

Future work should focus on improving the geometric accuracy of the depth map. One course of action worth investigating is to compute depth maps for each reference viewpoint (i.e. capturing camera), taking into account spatial and temporal constraints as explored by Kang et al. [2001]. The depth maps could then be combined into one depth map for the eye gaze corrected viewpoint, following the fusion principles of Merrell et al. [2007]. The confident-camera recoloring scheme could be improved by incorporating visibility and resolution constraints in a camera blending field, similar to the one used by unstructured lumigraph rendering [Buehler et al., 2001]. Alternatively, techniques from floating textures to color the geometry proxy could be investigated [Eisemann et al., 2008]. In the largest extent, the background could be interpolated with correct motion parallax. This would complete the immersive effect of a virtual window into the remote user's world.

Our prototype is a fully functional end-to-end system for close-up one-to-one eye gaze corrected video conferencing. It has a minimal amount of constraints, is intuitive to use and is very convincing as a proof-of-concept.

### 10.2.4 Immersive Collaboration Environment (Many Cameras)

Current tools for computer-supported cooperative work (CSCW) suffer from two major deficiencies [Otto et al., 2006; Wolff et al., 2007]. First, they do not allow to observe the body language, facial expressions and spatial context of the collaborators. Second, they miss the ability to naturally and synchronously manipulate objects in a shared environment.

In chapter 8, we solved these issues by integrating our view interpolation for eye gaze correction into an immersive environment that supports collaboration at a distance. We implemented the fundamental technical requirements of *The Office Of The Future* identified by Raskar et al. [1998], i.e. dynamic image-based modeling, rendering and a spatially immersive display, and extended them with cooperative surface computing and aural communication. Our final prototype, depicted in Figure 10.1(bottom-right), brings together many key research areas that have been the focus of our research group as a whole over the past years: view interpolation for free viewpoint video, calibration of camera networks, tracking, omnidirectional cameras, multi-projector immersive displays, multi-touch interfaces, and audio processing.

A good spatially immersive display must at least engulf the user's peripheral vision. We designed an easy to build and very affordable 180-degree surround display by spanning a matte white vinyl cloth over a truss constructed of lightweight aluminum bars. It is projected on by multiple calibrated and synchronized projectors that perform overlap intensity blending to seamlessly stitch the individual projections together. The projected image is geometrically corrected to compensate for the shape and any irregularities of the screen.

The user himself is captured by a sea of cameras that peek through cuts in the panoramic screen, but that are spaced far enough apart to not destroy the immersive experience. The curved nature of the panoramic screen implies that the cameras' centers of projection do not lie in the same plane and thus are not (stereo) rectified with respect to one another. Therefore, we opted for the same approach as our plane sweeping prototype to reconstruct the eye gaze corrected image. In fact, our plane sweeping algorithm can be carried over in its entirety and applied with minimal modifications, which in itself is great proof of its flexibility. Moreover, there is virtually no change in visual quality, though the larger user-to-screen distance implies that occlusions are less severe and artifacts are less noticeable (visual quality: 6/7, very good). The same larger distance also causes the eyes to be less clearly discernible, but nevertheless we could confirm that humans are very well adapted to picking up eye contact at even the farthest of distances (eye contact: 6/7, very good).

As convincing as our plane sweeping prototype is though, its main drawback is its lack of spatial context. This was originally welcomed as a means to avoid mismatches and thereby enhance the quality of the generated virtual imagery, but it poses a problem if our goal is to fully immerse the users in each other's environment. We tackled this by capturing the user's panoramic background using a Point Grey Ladybug3 omnidirectional camera. The result is an immersive environment that offers a wide and detailed view on the environment of the remote user, while being just as flexible in restoring and maintaining eye contact (spatial context: 7/7, excellent). At the same time, the local user's freedom to move in his own environment is restricted only by the size of the panoramic screen and the area that is covered by the capturing cameras (freedom of movement: 6/7, very good).

Arguably the largest drawback to our system is that it takes a lot of space, has the potential to be very expensive, and consists of many separate components that each need to be set up and maintained carefully. All properties that no doubt pose a challenge for the average user, yet are still more than conceivable in a professional office environment (physical complexity: 3/7, reasonable). Furthermore, making sure that all of the components operate synchronously does increase the system's processing complexity. However, in our current implementation each of them run on their own dedicated hardware and thus function independently of one another (algorithmic performance: 5/7, good).

We support collaboration through networked multi-touch surface computing. Our internally built hardware makes use of frustrated total internal reflection of infrared light to detect multiple touch points. On the software side, a framework is provided that allows files to be shared over the network and consequently be viewed, controlled, manipulated and annotated by both sides simultaneously, as if the users were actually working at the same surface. We discovered that the combination of multi-touch surfaces and immersive video conferencing is remarkably well-suited for the purpose of collaboration at a distance. The networked multi-touch surfaces allow multiple users to interact with the same application simultaneously, while directly manipulating their own private physical interface. Unlike with co-located collaboration, the users don't have to stand in front of the same table or screen and thereby risk getting in each other's way, while at the same time the immersive environment provides a very convincing sense of telepresence.

We currently developed our immersive environment with support for one-to-one communication and collaboration in mind. The system's spatial (large panoramic screen, lots of space to move around) and algorithmic (many virtual viewpoints can be reconstructed) nature has big potential to seamlessly support multiple users together, but a couple of issues will need to be addressed (communication modes: 6/7, very good).

First, we need the means to capture and model multiple users. To this end, we already proposed an adaptation of the plane sweeping algorithm (discussed further in section 10.5) to efficiently interpolate a complex scene that contains multiple dominant depths, e.g. when multiple users are present in our immersive environment. Second, individual users must be discerned and tracked, so that each can be offered their own viewpoint based on the position and orientation of their eyes. The tracker developed by Maesen et al. [2013] looks very applicable. It determines the six degrees of freedom of the head pose by passively tracking (infrared) LED strips mounted to the ceiling. It is independent of the size of the working area and puts no restriction on the number of users to track [Maesen, 2016]. Lastly, and most challenging, is showing a perspectively correct image to each user separately, from his own point of view, on the same panoramic screen. This is a problem that was well beyond the scope of this dissertation. A good place to start looking for solutions is the work of Nashel [2010], who trades off stereoscopic viewing in autostereoscopic parallax displays to provide unique monoscopic images to multiple viewers in different positions.

## 10.3 Maintaining Camera Calibration

As all of our prototypes capture images from live cameras, maintaining their calibration in a setting that is subject to a lot of dynamic user activity poses a challenge. Therefore, in chapter 3, we developed a novel algorithm to robustly restore the calibration of a camera network after the event of an (un-)intended camera displacement, without the need for a full system recalibration.

The algorithm works in two phases. In the first phase, displacement of a camera is detected by means of hotspots. Descriptive features in the background scene are chosen to be tracked in a temporal sequence of frames. A camera is deemed to have moved if a sufficient amount of its features no longer match. In the second phase, a moved camera is reinserted into the calibrated network by recomputing its geometric calibration from information provided by its calibrated neighbors. Recalibration is efficiently achieved using only image point correspondences with its neighbors, without the need to compute the 3D structure of the scene.

Currently, the main limitation of our algorithm is that it is unable to discern between a change in extrinsic and intrinsic camera parameters. If a change is detected, it automatically assumes that it has physically moved (i.e. its extrinsic parameters have changed), even though a change in intrinsic parameters might have occurred instead, e.g. a change in the zoom level.

## 10.4 Edge-Sensitive Disparity Estimation with Iterative Refinement

In chapter 5, we developed a novel disparity estimation algorithm that was then relied upon by our stereo interpolation prototype of chapter 6 to synthesize an eye gaze corrected image.

First, we presented a matching cost aggregation method that uses two edge-sensitive shape-adaptive support windows per pixel neighborhood; one following the horizontal edges in the image, the other the vertical edges. Together they form the final aggregation window shape that closely follows all object edges and thereby achieves increased disparity hypothesis confidence.

Second, we formalized a four-stage iterative disparity refinement process that relies on the same support windows covering image patches of similar color. By assuming that color discontinuity boundaries in the image are also depth discontinuity boundaries in the scene, the refinement is able to efficiently detect and fill in occlusions. As the only prior knowledge it needs are the input color images, it can be applied to any initially estimated disparity map. Furthermore, the iterative process quickly converges to a final solution.

We designed our stereo matching algorithm for efficient implementation in CUDA, which exposes the GPU as a directly operable massive pool of parallel threads. Rectified stereo matching processes pixels scanline by scanline, which maps perfectly to the coalesced global memory access requirements of CUDA and avoids lots of memory transfer overhead. As a result, our algorithm executes in real-time, is easy to understand and implement and generates smooth disparity maps with sharp object edges and little to no artifacts. It is very competitive with the current state-of-the-art of real-time local stereo matching algorithms.

We currently consider the weakest link to be the rather rudimentary pixel-wise color consistency check to determine the extent of the support windows. Relying on more precise color-based image segmentation has the potential to increase the matching quality considerably. Accurate segmentation to consider is mean-shift segmentation [Comaniciu and Meer, 2002; Gerrits and Bekaert, 2006] and superpixels [Zitnick and Kang, 2007]. In the context of video conferencing, it is also especially worth considering incorporating temporal information [Davis et al., 2005]. By extending our shape-adaptive planar windows to volumetric grids in the temporal domain, we would be able to process over successive video frames.

## 10.5 Controlling Algorithmic Complexity

Throughout this dissertation, we have developed several dynamic control mechanisms that decrease the algorithmic complexity of stereo matching and plane sweeping, while increasing view synthesis quality. The control mechanisms are all based on a histogram analysis of the concerning disparity map or depth map. They were developed for stereo matching by hierarchically restricting the disparity range (in section 6.2, p. 105), for single object-of-interest plane sweeping by redistributing the planes uniformly in a dynamically shifting depth range (in section 7.2.5, p. 142) and for full-scene plane sweeping by redistributing the planes non-uniformly in a fixed depth range (in section 8.3, p. 170).

For stereo matching, instead of trying to improve the cost aggregation or disparity refinement stages, as most of the literature focuses on, we introduced the idea of limiting the disparity range itself. In a two-pass process, the images are first matched over the full disparity search range, but at a reduced (downsampled) resolution. Peaks in the disparity map's histogram then indicate where objects are located in the scene, since many pixels will possess that peak's disparity in question. By excluding all histogram bins (i.e. disparity hypotheses) that are below a given threshold (proportional to the image resolution or the histogram entropy), the disparity search range is effectively restricted during a second pass over the full image resolution. The complexity of the histogram computation is relatively low and in turn the potential to accelerate the local stereo matching algorithm becomes very high.

For plane sweeping of scenes with a single dominant depth (e.g. the user's head and torso), we devised a method to efficiently keep a uniform distribution of planes focused around a single object-of-interest. After fitting a Gaussian distribution on the histogram of the depth map, its mean and standard deviation will indicate the depth and extent of the object. When the object moves, the mean and standard deviation will change and we can use this to retroactively adapt the effective depth range by updating the minimum and maximum plane depths accordingly. The result is a uniform plane distribution, narrowly focused around the object-of-interest, that responds to movements of the object in the scene by dynamically shifting back and forth.

For plane sweeping of complex scenes with multiple dominant depths (e.g. multiple users and a background), we devised a method to non-uniformly distribute the planes according to the scene content. This time, we interpret the cumulative histogram of the depth map as a probability density function that describes the likelihood that a plane should be placed at a particular depth in the scene. In other words, the planes are rearranged to be more densely packed in regions with many objects and to be more sparsely present in regions with few to no objects. The result is an adaptive non-uniform plane distribution that responds to a redistribution of any and all content in the scene.

All of these approaches decrease computational complexity, while increasing the quality by implicitly reducing the chance at mismatches. Although developed in the context of video conferencing, they are generally applicable to all kinds of scenes and scenarios.

One key observation points in the direction of future work. On the one hand, the minimum and maximum plane depths of the dynamic uniform plane distribution move back and forth, with planes uniformly distributed in between, to leverage the dynamic range. On the other hand, the minimum and maximum plane depths of the adaptive non-uniform plane distribution remain fixed, with planes non-uniformly distributed and moving in between, to keep the range from collapsing while adapting to changing scene content. A dynamic adaptive non-uniform distribution would combine the best of both worlds, leveraging the dynamic range while adapting to the entire scene.

# Appendix A

# Questionnaires

Table A.1 and Table A.2 below relate to the evaluation of several video conferencing technologies carried out in chapter 9.

Table A.1 contains the questionnaire that was filled out by the participants after testing each of the four technology use cases from section 9.2.1, p. 187. Originally drafted in Dutch, it has here been translated verbatim to English for convenience.

**Table A.1:** Questionnaire filled out after using a video conferencing setup.

A number of claims about the system that you have just worked with will now follow. Underneath them it says *completely disagree* to the left, and *completely agree* to the right. In between there are 7 circles. By checking one of these circles, you indicate to what degree you agree with the claim:

Completely disagree ○ ⊗ ○ ○ ○ ○ ○ Completely agree

**1. This system invites to engage in a conversation spontaneously.**
Completely disagree ○ ○ ○ ○ ○ ○ ○ Completely agree

**2. During the use of this system, it felt as if the other person was physically present.**
Completely disagree ○ ○ ○ ○ ○ ○ ○ Completely agree

**Table A.1:** Questionnaire filled out after using a video conferencing setup. (continued)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **3.** | **This system is fitting for communication in a social context, e.g. with friends, acquaintances or family.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **4.** | **This system is fitting for communication in a professional context, e.g. in companies.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **5.** | **I thought that my conversation partner came across lifelike.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **6.** | **Speaking in turns went well.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **7.** | **It's pleasant to use this system to talk to someone.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **8.** | **It's pleasant to use this system to listen to someone.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **9.** | **It's pleasant to use this system to see someone.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **10.** | **I had the feeling there was good and effective communication.** | | | | | | | | | |
| | Completely disagree | ○ | ○ | ○ | ○ | ○ | ○ | ○ | Completely agree | |
| **11.** | **Try to indicate the image quality of this system (1 = low, 10 = high).** | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **12.** | **Try to indicate the life-likeness of this system (1 = low, 10 = high).** | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **13.** | **Try to indicate how intimate this system feels (1 = low, 10 = high).** | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **14.** | **Try to indicate how pleasant you've found this form of communication (1 = low, 10 = high).** | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

The filled out questionnaires were analyzed by fitting a GLM (generalized linear model). In Table A.2, the results are reproduced verbatim from the original report. For each question the point scale (e.g. 7-point) and resulting mean (M) and standard deviation (SD) are reported. The UCx headings respectively refer to the single webcam (UC1), our face-to-face prototype (UC2), ETRO's enhanced reality (UC3) and ETRO's animated avatar (UC4) use cases as described in section 9.2.1, p. 187. For reasons explained in section 9.2.2, p. 188, only a limited number of participants were able to evaluate ETRO's enhanced reality (UC3) setup. Consequently, to be able to apply meaningful statistics, the other three use cases are first reported on. The scores of UC3 are still printed in gray font, but not included in the main analysis, since this is a within subjects design and it would considerably bring down the number of participants that used all four systems. For the detailed data and GLM analysis, please refer to the work by CUO [van Nimwegen, 2008].

**Table A.2:** Questionnaires scores.

| | Scores | | | |
|---|---|---|---|---|
| | **UC1** | **UC2** | UC3 | **UC4** |
| **Question 1** *(7-point scale)* This system invites to engage in a conversation spontaneously. | M = 5.33 SD = 1.20 | M = 5.02 SD = 1.35 | M = 3.92 SD = .5 | M = 2.79 SD = 1.43 |
| GLM showed that there was a significant overall difference F(2,82) = 93.94, p < .001 between the systems. Post-hoc comparisons showed that the difference between setups 1-2 did not differ, but cores on system 4 were significantly lower than on 1 and 2, p < .001. When the enhanced reality system was included, post-hoc comparison shows that the scores differ significantly with all the other systems (p < .05). | | | | |
| **Question 2** *(7-point scale)* During the use of this system, it felt as if the other person was physically present. | M = 4.62 SD = 1.46 | M = 4.12 SD = 1.52 | M = 4.33 SD = 1.76 | M = 2.29 SD = 1.37 |
| GLM showed that there was a significant overall difference F(2,82) = 27.98, p < .001 between the systems. Post-hoc comparisons showed all the three systems differed significantly (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores differ significantly only with system 4 (p < .001). | | | | |

**Table A.2:** Questionnaires scores. (continued)

| | Scores | | | |
|---|---|---|---|---|
| | **UC1** | **UC2** | UC3 | **UC4** |
| **Question 3** *(7-point scale)* This system is fitting for communication in a social context, e.g. with friends, acquaintances or family. | M = 5.57 SD = 1.5 | M = 5.17 SD = 1.68 | M = 4.25 SD = 2.05 | M = 2.36 SD = 1.45 |
| GLM showed that there was a significant overall difference F(2,82) = 115.71, p < .001 between the systems. Post-hoc comparisons showed all the three systems differed significantly (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores differ significantly with system 1 and 4 (p < .05). | | | | |
| **Question 4** *(7-point scale)* This system is fitting for communication in a professional context, e.g. in companies. | M = 5.31 SD = 1.42 | M = 5.00 SD = 1.67 | M = 4.76 SD = 1.78 | M = 2.19 SD = 1.50 |
| GLM showed that there was a significant overall difference F(2,82) = 70.65, p < .001 between the systems. Post-hoc comparisons showed that system 1-2 did not differ, but that 4 differed from both 1-2 (p < .001). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with system 1 and 4 (p < .05). | | | | |
| **Question 5** *(7-point scale)* I thought that my conversation partner came across lifelike. | M = 5.29 SD = 1.35 | M = 4.93 SD = 1.61 | M = 4.50 SD = 1.88 | M = 1.76 SD = .93 |
| GLM showed that there was a significant overall difference F(2,82) = 125.41, p < .001 between the systems. Post-hoc comparisons showed that again, system 1-2 did not differ, but that 4 differed from both 1-2 (p < .001). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly only with the score of 4 (p < .001). | | | | |
| **Question 6** *(7-point scale)* Speaking in turns went well. | M = 6.00 SD = .76 | M = 5.93 SD = .89 | M = 5.00 SD = 1.35 | M = 4.40 SD = 1.2 |
| GLM showed that there was a significant overall difference F(2,82) = 50.12, p < .001 between the systems. Post-hoc comparisons showed that again, system 1-2 did not differ, but that 4 differed from both 1-2 (p < .001). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with the scores of 1 and 2 (p < .05). | | | | |

**Table A.2:** Questionnaires scores. (continued)

| | Scores | | | |
|---|---|---|---|---|
| | **UC1** | **UC2** | UC3 | **UC4** |
| **Question 7** *(7-point scale)* It's pleasant to use this system to talk to someone. | M = 5.17 SD = 1.32 | M = 4.88 SD = 1.64 | M = 4.17 SD = 2.04 | M = 2.36 SD = 1.28 |
| GLM showed that there was a significant overall difference F(2,82) = 84.73, p < .001 between the systems. Post-hoc comparisons showed that again, system 1-2 did not differ, but that 4 differed from both 1-2 (p < .001). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with only the score on system 4 (p < .05). | | | | |
| **Question 8** *(7-point scale)* It's pleasant to use this system to listen to someone. | M = 5.33 SD = 1.14 | M = 5.14 SD = 1.48 | M = 4.58 SD = 1.51 | M = 3.38 SD = 1.56 |
| GLM showed that there was a significant overall difference F(2,82) = 40.17, p < .001 between the systems. Post-hoc comparisons showed that again, system 1-2 did not differ, but that 4 differed from both 1-2 (p < .001). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with only the score on system 4 (p < .05). | | | | |
| **Question 9** *(7-point scale)* It's pleasant to use this system to see someone. | M = 5.55 SD = 1.09 | M = 4.76 SD = 1.76 | M = 4.17 SD = 2.13 | M = 1.67 SD = .90 |
| GLM showed that there was a significant overall difference F(2,82) = 121.81, p < .001 between the systems. Post-hoc comparisons showed that all the measured differences were significant (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with the score on systems 1 and 4 (p < .05). | | | | |
| **Question 10** *(7-point scale)* I had the feeling there was good and effective communication. | M = 5.64 SD = .91 | M = 5.31 SD = 1.22 | M = 4.92 SD = 1.24 | M = 3.90 SD = 1.36 |
| GLM showed that there was a significant overall difference F(2,82) = 42.35, p < .001 between the systems. Post-hoc comparisons showed that all the measured differences were significant (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with only the score on system 4 (p < .05). | | | | |

**Table A.2:** Questionnaires scores. (continued)

| | Scores | | | |
|---|---|---|---|---|
| | **UC1** | **UC2** | UC3 | **UC4** |
| **Question 11** *(10-point scale)* Try to indicate the image quality of this system. | M = 7.50 SD = 1.74 | M = 6.19 SD = 2.05 | M = 5.42 SD = 2.64 | M = 3.40 SD = 1.99 |
| GLM showed that there was a significant overall difference F(2,82) = 83.13, p < .001 between the systems. Post-hoc comparisons showed that all the measured differences were significant (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with the score on systems 1 and 4 (p < .05). | | | | |
| **Question 12** *(10-point scale)* Try to indicate the life-likeness of this system. | M = 7.17 SD = 1.53 | M = 6.26 SD = 2.13 | M = 5.75 SD = 2.70 | M = 2.38 SD = 1.53 |
| GLM showed that there was a significant overall difference F(2,82) = 139.40, p < .001 between the systems. Post-hoc comparisons showed that all the measured differences were significant (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with the score on systems 1 and 4 (p < .05). | | | | |
| **Question 13** *(10-point scale)* Try to indicate how intimate this system feels. | M = 7.29 SD = 1.60 | M = 6.43 SD = 2.19 | M = 6.33 SD = 2.35 | M = 3.07 SD = 1.72 |
| GLM showed that there was a significant overall difference F(2,82) = 108.17, p < .001 between the systems. Post-hoc comparisons showed that all the measured differences were significant (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with the score on systems 1 and 4 (p < .05). | | | | |
| **Question 14** *(10-point scale)* Try to indicate how pleasant you've found this form of communication. | M = 7.61 SD = 1.50 | M = 6.44 SD = 2.34 | M = 6.33 SD = 2.61 | M = 2.93 SD = 1.96 |
| GLM showed that there was a significant overall difference F(2,82) = 113.93, p < .001 between the systems. Post-hoc comparisons showed that all the measured differences were significant (p < .05). When the enhanced reality system was included, post-hoc comparison shows that the scores of system 3 differ significantly with the score on systems 1 and 4 (p < .05). | | | | |

# Nederlandstalige Samenvatting

De gebruikelijke videoconferentie-opstelling (denk bijvoorbeeld aan Skype met een webcam) vertoont een aantal fundamentele gebreken die het emuleren van een waar gesprek in persoon in de weg staan; het ontbeert als het ware een gevoel van samenzijn met en onderdompeling in de wereld van de gesprekspartner. Een prominent probleem is het gebrek aan direct oogcontact tussen de gesprekspartners. De webcam van de gebruiker is immers opgesteld naast het scherm of in het beste geval geïntegreerd in de rand ervan. Hierdoor is de gebruiker genoodzaakt zijn blik te wisselen tussen het observeren van zijn gesprekspartner op het scherm en het staren in zijn eigen webcam. Het is dat conflict tussen beide kijkrichtingen dat het maken van oogcontact belemmert. Deze kwestie van ontbrekend oogcontact is het centraal op te lossen probleem in dit proefschrift.

**Een Beeldgebaseerde Aanpak** Voor het oplossen van ons probleem kiezen we voor een beeldgebaseerde aanpak, waarmee we bedoelen dat we een nieuw beeld synthetiseren uit beelden die (live) door camera's opgenomen worden. In het nieuw berekende beeld zal de blik van de gebruiker gecorrigeerd zijn en het conflict tussen de kijkrichtingen bijgevolg niet meer bestaan. Door te werken met live opgenomen beelden vermijden we de kunstmatige uitstraling van veel eerdere oplossingen die trachten de gebruiker geometrisch te reconstrueren of te belichamen met een avatar. Voornamelijk onderzoeken we drie verschillende beeldsynthese-algoritmen om ons doel te bereiken, wat resulteert in concrete bijdragen aan omnidirectionele omgevingsopname (environment mapping), aan dispariteitsschatting uit gerectificeerde stereobeelden (stereo matching) en aan vlakvegen (plane sweeping).

Al onze algoritmen ontwerpen en implementeren we uitsluitend voor en op de grafische kaart. We buiten met andere woorden de grafische kaart uit voor algemene – niet-grafische – berekeningen, waardoor we profiteren van haar enorme geparallelliseerde rekenkracht en we ons verzekeren van een ogenblikkelijke verwerking en toekomstbestendige schaalbaarheid.

Hoewel we ze aanwenden voor blikcorrectie, zijn onze algoritmen algemeen toepasbaar op een grotere verscheidenheid aan scènes en gebruiksscenario's.

**Vier Prototypes** We ontwerpen vier verschillende prototypes die elk de blik van de gebruiker op hun eigen manier corrigeren. Elk prototype doet beroep op een specifiek beeldsynthese-algoritme (of een combinatie van algoritmen) en elk beeldsynthese-algoritme steunt op zijn beurt op een specifieke configuratie van camera's voor het aanleveren van zijn invoerbeelden. Bijgevolg kunnen we de prototypes consistent rangschikken volgens toenemende fysieke complexiteit van de cameraopstelling.

**Handhaven van Camerakalibratie** We bouwen onze prototypes met het oog op daadwerkelijk gebruik in de praktijk. Dat vergemakkelijkt het handhaven van de kalibratie van hun camera's zeker niet, aangezien een camera – al dan niet onbedoeld – al snel wordt bewogen. Daarom bedenken we eerst een efficiënt algoritme om de beweging van een camera te detecteren en vervolgens de bewogen camera te re-integreren in het netwerk van reeds gekalibreerde camera's.

Als we veronderstellen dat de intrinsieke kalibratie van de bewogen camera niet verloren is gegaan (beweging speelt zich immers af in de extrinsieke parameters), dan kunnen we haar nieuwe extrinsieke kalibratie als volgt bepalen. Eerst berekenen we een essentiële matrix tussen de bewogen camera en elk van haar naburige camera's aan de hand van puntcorrespondenties in de beelden. Dat geeft ons een schatting van een lokaal coördinatenstelsel voor elk camerapaar, waarbij elk lokaal coördinatenstelsel gerelateerd is aan het gemeenschappelijke wereldcoördinatenstelsel volgens een gelijkvormigheidstransformatie. Uit alle lokale schattingen leiden we tenslotte een (gemiddelde) rotatie en (intersecterende) translatie af die samen de nieuwe extrinsieke kalibratie vormen in het wereldcoördinatenstelsel van het voorheen volledig gekalibreerde systeem.

In tegenstelling tot andere herkalibratietechnieken herconstrueren we niet expliciet de 3D structuur van de scène, maar hebben we enkel nood aan puntcorrespondenties in het beelddomein. We behalen accurate resultaten met een herprojecteringsfout van minder dan een pixel. Ons algoritme is dan ook competitief met de actuele stand van de technologie inzake herkalibratie van ge(-de-)centraliseerde cameranetwerken.

**Prototype 1: Herschikken van Omnidirectionele Omgevingsopname** Ons eerste prototype is ook meteen onze meest buiten-de-lijntjes-kleurende oplossing. Het vereist slechts één enkel camerabeeld als invoer, tezamen met een projector voor de weergave van het gecorrigeerde beeld. We nemen een omnidirectioneel beeld (met andere woorden, de omgeving) op door een (noordelijk) halfronde spiegel te filmen en combineren dit – na herstructurering van het gefilmde beeld – met projectie op een (zuidelijk) halfrond mat scherm. Beide halfronden vormen samen één enkele complete bol, als het ware één enkel communicatieapparaat met een opnamefunctie bovenaan en een weergavefunctie onderaan.

De onconventionele nieuwigheid zit hem in het feit dat we geen beeldinterpolatie uitvoeren in de strikte zin van het woord, maar in plaats daarvan een gecorrigeerd beeld synthetiseren door de omgevingsopname pixel-per-pixel te herschikken. We gaan uit van parallelle lichtstralen onder een affien cameramodel en ontwikkelen zo de wiskundige vergelijkingen die uitdrukken welke pixel in het invoerbeeld moet worden afgebeeld op welke pixel in het uitvoerbeeld. De resulterende vergelijkingen zijn volledig onafhankelijk van de structuur van de scène en maken diepteschatting overbodig. Bijgevolg hoeven ze slechts eenmaal vooraf worden uitgewerkt, waarna de beeldtransformatie kan ondergebracht worden in een opzoekingstabel. Dat maakt het algoritme niet alleen uitermate geschikt voor implementatie op de grafische kaart, maar laat zelfs een zeer efficiënte executie door de CPU (centrale verwerkingseenheid) toe.

Het ontvouwen van de omgevingsreflectie op een (relatief kleine) spiegelende bol, levert omnidirectionele beelden op met een projectiecentrum in het midden van die bol. Bijgevolg staart de gebruiker rechtstreeks in de camera wanneer hij kijkt naar het middelpunt van de bol en is oogcontact inherent gewaarborgd. Bovendien ondersteunt het concept moeiteloos vele gebruikers tegelijkertijd, onthult het hun volledige ruimtelijke context en biedt het hen een ongekende vrijheid van beweging. Het grootste nadeel is echter de beeldkwaliteit. Deze wordt ernstig beperkt door het mathematische model en de ruwe hardwarecomponenten.

**Randgevoelige Dispariteitsschatting met Iteratieve Verfijning** Aan de basis van ons tweede prototype, dat we zo meteen zullen voorstellen, ligt ons inventief lokaal dispariteitsschattingsalgoritme. We leveren drie voorname bijdragen.

Ten eerste presenteren we een nieuwe methode voor vergelijkingskostaggregatie die twee randgevoelige aggregatievensters per pixelbuurt gebruikt. De vensters zijn zo gedefinieerd dat ze gebieden met gelijkaardige kleur in het beeld bedekken. Het ene venster past zich aan aan de horizontale randen in het beeld, het andere aan de verticale. Samen vormen ze het finale aggregatievenster dat nauwgezet de randen van een object in de scène traceert, wat resulteert in een nauwkeurigere vergelijkingskost (en dus dispariteitsschatting).

Ten tweede ontwerpen we een iteratief proces om het berekende dispariteitsbeeld te verfijnen. Het proces bestaat uit vier strikt geformaliseerde fasen (kruis-controle, bitgewijs stemmen, ongeldige dispariteiten afhandelen, mediaanfilteren) en steunt voornamelijk op dezelfde horizontale en verticale vensters. Door te veronderstellen dat de kleurranden in het beeld ook de dieptediscontinuïteiten van objecten in de scène zijn, is het verfijningsproces in staat om efficiënt occlusies op te sporen en in te vullen. Het proces vereist enkel de kleurbeelden als voorkennis, kan worden toegepast op elk initieel geschat dispariteitsbeeld en convergeert snel naar een eindoplossing.

Naast het verbeteren van de kostaggregatie en dispariteitsverfijning, introduceren we tenslotte het idee om het zoekbereik van dispariteitshypothesen zelf te beperken. We merken eerst op dat pieken in het histogram van het dispariteitsbeeld aangeven op welke diepte objecten zich bevinden in de scène (elke kolom in het histogram komt overeen met een bepaalde dispariteitshypothese). Daarentegen wordt ruis in het histogram met hoge waarschijnlijkheid veroorzaakt door foutieve dispariteitsschattingen. Hieruit leiden we de volgende hiërarchische procedure af. Eerst berekenen we het dispariteitsbeeld en zijn histogram op een lagere resolutie van de invoerbeelden. Vervolgens worden alle dispariteitshypothesen waarvoor de kolomwaarde onder een dynamisch bepaalde drempelwaarde (evenredig aan de beeldresolutie of de histogram-entropie) blijft, uitgesloten uit het zoekbereik voor de oorspronkelijke (hogere) resolutie. Het opstellen van het lageresolutie histogram is relatief goedkoop, wat ons de kans biedt om in één klap de kwaliteit van het dispariteitsbeeld te verhogen en de verwerkingscomplexiteit van het algoritme te verlagen. Bovendien is deze strategie toepasbaar op elk lokaal dispariteitsschattingsalgoritme.

Voor de implementatie kiezen we voor de moderne programmeertaal CUDA van NVIDIA. CUDA legt de hardware paradigmatisch bloot als een enorme verzameling van direct aanspreekbare parallelle threads (verwerkingsdraden), wat zeer goed aansluit bij de algoritmische structuur van beeldlijn-gerectificeerde pixelgewijze dispariteitsschatting. Op hedendaagse hardware kunnen we een verwerkingssnelheid van ongeveer 12 FPS bereiken voor de standaardresolutie ($450 \times 375$) van de Middlebury databank.

Ons algoritme is eenvoudig te begrijpen en te implementeren. Het genereert egale dispariteitsbeelden met scherpe randen en weinig tot geen storende artefacten. Het is zeer concurrentieel met de actuele ontwikkelingen inzake lokale dispariteitsschattingsalgoritmen.

**Prototype 2: Stereo Interpolatie** Voor ons tweede prototype doen we een beroep op gerectificeerde stereo interpolatie. We monteren een camera links en rechts van het scherm en laten de gebruiker in het horizontale midden plaatsnemen. We interpoleren vervolgens het tussenliggende (en dus blikgecorrigeerde) camerastandpunt, waarbij we het stappenplan voor depth-image-based rendering (DIBR, dieptebeeldgebaseerde beeldsynthese) volgen en uitbreiden. Dat stappenplan bestaat in wezen uit een dispariteitsschattingsfase en een beeldsynthesefase. De beeldsynthesefase is vrij eenvoudig en zeer efficiënt, maar leunt sterk op accurate dispariteitsschatting om de pixels van de invoerbeelden correct te kunnen verplaatsen naar het tussenliggende standpunt.

Enerzijds is het tweede prototype dankzij ons accuraat dispariteitsschattingsalgoritme in staat om een beeld te synthetiseren waarin de gecorrigeerde blik van de gebruiker zeer scherp en duidelijk waarneembaar is. Anderzijds is deze afhankelijkheid ook de grootste handicap, omwille van twee redenen. Ten eerste wordt de bewegingsvrijheid van de gebruiker gelimiteerd tot de horizontale basislijn tussen de linker- en rechtercamera. Bijgevolg valt oogcontact moeilijk te handhaven. Ten tweede geven algoritmen voor de schatting van dichte dispariteitsbeelden de voorkeur aan camera's in een opstelling met een kleine basis. Ze trachten zo occlusies te minimaliseren en een optimaal resultaat te bekomen. In de praktijk dwingt dat ons echter ofwel de camera's rond een kleiner scherm te plaatsen, ofwel een grotere afstand tussen de gebruiker en het scherm te verplichten.

**Prototype 3: Vlakvegen** Met ons derde prototype streven we ernaar de tekortkomingen van stereo interpolatie te overwinnen door zes camera's nauw rond het scherm te monteren op een op maat gemaakt lichtgewicht metalen kader. Deze algemenere cameraopstelling vermijdt al te grote occlusies, maar aangezien gerectificeerde stereo niet overweg kan met een dergelijke opstelling, moeten we ons wenden tot vlakvegen om de blik van de gebruiker te corrigeren. Het flexibelere vlakvegen biedt ons meer vrijheid in het kiezen van het te reconstrueren standpunt, zonder dat we daarbij noodzakelijk aan beeldextrapolatie hoeven te doen. In combinatie met gelijktijdige oogtracering om het gezichtspunt van de gebruiker te bepalen, zorgt dat ervoor dat oogcontact ten allen tijde en vanaf elke positie en hoek kan behouden blijven.

Een aantal weldoordachte ontwerp- en implementatiekeuzes resulteren in een optimaal presterend systeem. Ze laten ons toe een verwerkingssnelheid van meer dan 40 FPS voor de SVGA-resolutie ($800 \times 600$) te bereiken zonder merkbaar verlies van de visuele kwaliteit, zelfs op grafische kaarten uit het lagere marktsegment.

Onze eerste optimalisatie vertrekt vanuit onze strategie om het zoekbereik bij dispariteitsschatting te beperken. We deduceren een gelijkaardige manier om op efficiënte wijze een uniforme verdeling van vlakken gegroepeerd te houden rond één enkel dominant object – m.a.w. het hoofd en de torso van de gebruiker – dat zich door de scène beweegt. Een Gaussverdeling, gefit over het histogram van het dieptebeeld, zal de diepte (gemiddelde) en de omvang (standaardafwijking) van het object in kwestie aangeven. We kunnen dit gebruiken om met terugwerkende kracht te reageren op bewegingen van het object door een compactere set van vlakken heen en weer te schuiven, in plaats van voortdurend de volledige werkruimte af te tasten met een schaarsere distributie. Deze strategie verlaagt niet alleen de algoritmische complexiteit van vlakvegen, maar verhoogt impliciet ook de nauwkeurigheid ervan door de kans op ongeldige diepteschattingen aanzienlijk te verminderen.

Ten tweede presenteren we een iteratieve spatiale filter die fotometrische artefacten uit het geïnterpoleerde kleurenbeeld verwijdert door geometrische uitschieters in het bijhorende dieptebeeld – dat verondersteld wordt lokaal lineair te zijn – te detecteren en te corrigeren.

Ten derde kiezen we deze keer voor een implementatie in OpenGL en Cg om de vertex- en fragmentverwerking van de traditionele weergaveprocedure van de grafische kaart te herprogrammeren. Deze werkwijze komt beter overeen met de algoritmische structuur en de inherent willekeurige geheugentoegang van vlakvegen. Bovendien berust de traditionele weergaveprocedure op verwerking van polygonen, wat ons toelaat om granulaire optimalisatiestrategiën te introduceren en daardoor de prestaties nog verder te verbeteren.

Tenslotte stellen we een goed gedefinieerde set van parameters samen die het systeem een gebruikersonafhankelijke toepasbaarheid verleent. Het resultaat is een overtuigend en volledig functioneel prototype dat videoconferenties tussen twee gesprekspartners van dichtbij ondersteunt, dat een minimaal aantal beperkingen heeft en dat intuïtief is in gebruik.

**Prototype 4: Immersieve Samenwerkingsomgeving** Voor de realisatie van ons vierde en laatste prototype vertrekken we vanuit de erkenning dat de huidige instrumenten voor computer-gemedieerd coöperatief werken (CSCW, computer-supported cooperative work) op twee belangrijke punten tekortschieten. Ten eerste laten ze niet toe de lichaamstaal, gezichtsuitdrukkingen en ruimtelijke context van de medewerkers (op afstand) waar te nemen. Ten tweede missen ze de mogelijkheid om op een natuurlijke en synchrone manier objecten te manipuleren in een gedeelde omgeving.

We lossen deze problemen op door ons vlakveegalgoritme te integreren in een immersieve omgeving die samenwerking op afstand ondersteunt. We identificeren en implementeren daarbij vijf fundamentele technische vereisten voor de ultieme samenwerkingsomgeving, namelijk dynamische beeldgebaseerde modellering, daaropvolgende reconstructie en correctie voor weergave, een immersief beeldscherm, een coöperatief multi-aanraakscherm, en auditieve communicatie.

We introduceren ook een laatste aanpassing aan het vlakveegalgoritme, ditmaal om complexe scènes met meerdere dominante objecten – m.a.w. wanneer meerdere gebruikers in de immersieve omgeving aanwezig zijn – efficiënt te interpoleren. Daartoe interpreteren we het cumulatieve histogram van het dieptebeeld als een kansdichtheidsfunctie die de kans uitdrukt dat een vlak op een bepaalde diepte in de scène moet geplaatst worden. Zo bekomen we een niet-uniforme verdeling van de vlakken die reageert op een herverdeling van alle inhoud van de scène.

Ons laatste prototype brengt vele kerndomeinen waar ons onderzoeksinstituut zich de afgelopen jaren op gericht heeft werkelijk samen: beeldinterpolatie, kalibratie van cameranetwerken, tracking, omnidirectionele camera's, multi-projector immersieve beeldschermen, multi-aanraak interfaces, en audioverwerking.

**Zeven Geëvalueerde Vereisten** Praktijkervaring met onze prototypes brengt aan het licht dat er naast oogcontact nog andere factoren bijdragen aan het beleven van een waar gevoel van samenzijn en immersie in de wereld van de gesprekspartner. We zien zeven vereisten voortdurend terugkeren en identificeren ze als: een gecorrigeerde blik (en het daaruit voortkomend oogcontact), het kunnen observeren van de ruimtelijke context van de gesprekspartner, zich vrij kunnen bewegen in de eigen ruimte, de beeldkwaliteit, de algoritmische prestaties, de fysieke complexiteit van de opstelling, en de ondersteunde communicatiemodi (één-op-één, veel-op-veel, meerdere partijen). We stellen empirisch vast dat deze vereisten onderworpen zijn aan vele afwegingen en onderlinge afhankelijkheden, naargelang we onze prototypes vergelijken en ze (informeel) evalueren in functie ervan.

Tenslotte wijst een beknopte gebruikersstudie niet alleen op het belang van de zeven vereisten, maar bekrachtigt ze ook onze initiële keuze voor beeldgebaseerde methoden. Om echter te komen tot de ideale videoconferentie-oplossing, zou er meer inzicht moeten verkregen worden in wat het betekent om virtueel samenzijn te ervaren en welke factoren deze ervaring precies mogelijk maken. Toch zijn wij van mening dat de zeven vereisten een referentiekader voorzien rond de ervaring die is opgedaan in dit proefschrift en dat ze kunnen dienen om toekomstige oplossingen voor videoconferentie met gecorrigeerde blik te ontwerpen, ontwikkelen en evalueren.

# Scientific Publications

This appendix lists all the author's publications, contributed to the scientific field and related to this dissertation.

## Related Scientific Publications

**[Hermans et al., 2006]** Chris Hermans, Maarten Dumont, Philippe Bekaert, and Eric Joris. JanusLights: A Camera-Projection System for Telematic Omni-Presence with Correct Eye Gaze. In *Proceedings of the 3rd IEEE International Workshop on Projector-Camera Systems (PROCAMS '06)*, pages 5–6, New York, NY, USA, June 2006

**[Hermans et al., 2007a]** Chris Hermans, Maarten Dumont, and Philippe Bekaert. JanusLights: A Camera-Projection System for Telematic Omni-Presence with Correct Eye Gaze. In *Proceedings of the IX Symposium on Virtual and Augmented Reality (SVR '07)*, Petrópolis, Rio de Janeiro, Brazil, May 2007a
- Won the *Best Paper* award.

**[Hermans et al., 2007b]** Chris Hermans, Maarten Dumont, and Philippe Bekaert. Extrinsic Recalibration in Camera Networks. In *Proceedings of the 4th Canadian Conference on Computer and Robot Vision (CRV '07)*, pages 3–10, Montreal, QC, Canada, May 2007b

**[Dumont et al., 2008]** Maarten Dumont, Steven Maesen, Sammy Rogmans, and Philippe Bekaert. A Prototype for Practical Eye-Gaze Corrected Video Chat on Graphics Hardware. In *Proceedings of the 5th International Conference on Signal Processing and Multimedia Applications (SIGMAP '08)*, pages 236–243, Porto, Portugal, July 2008
- Won the *Best Student Paper* award.

**[Dumont et al., 2009a]** Maarten Dumont, Sammy Rogmans, Gauthier Lafruit, and Philippe Bekaert. Immersive Teleconferencing with Natural 3D Stereoscopic Eye Contact using GPU Computing. In *Proceedings of the 2009 3D Stereo Media Conference (3DSM '09)*, Liège, Belgium, December 2009a

- Won the *Best Paper* award.

**[Dumont et al., 2009b]** Maarten Dumont, Sammy Rogmans, Steven Maesen, and Philippe Bekaert. Optimized Two-Party Video Chat with Restored Eye Contact using Graphics Hardware. In Joaquim Filipe and Mohammad S. Obaidat, editors, *e-Business and Telecommunications*, volume 48 of *Communications in Computer and Information Science (CCIS)*, pages 358–372. Springer Berlin Heidelberg, November 2009b

**[Rogmans et al., 2009a]** Sammy Rogmans, Maarten Dumont, Tom Cuypers, Gauthier Lafruit, and Philippe Bekaert. Complexity Reduction of Real-Time Depth Scanning on Graphics Hardware. In *Proceedings of the 4th International Conference on Computer Vision Theory and Applications (VISAPP '09)*, pages 547–550, Lisbon, Portugal, February 2009a

**[Rogmans et al., 2009b]** Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Migrating Real-Time Image-Based Rendering from Traditional to Next-Gen GPGPU. In *Proceedings of the 2009 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, pages 1–4, Potsdam, Germany, May 2009b

**[Dumont et al., 2010]** Maarten Dumont, Steven Maesen, Karel Frederix, Chris Raymaekers, Philippe Bekaert, and Frank Van Reeth. Immersive Collaboration Environment. In Elisabetta Di Nitto and Ramin Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science (LNCS)*, pages 187–188. Springer Berlin Heidelberg, 2010

**[Rogmans et al., 2010a]** Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Biological-Aware Stereoscopic Rendering in Free Viewpoint Technology using GPU Computing. In *Proceedings of the 2010 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON '10)*, pages 1–4, Tampere, Finland, June 2010a

**[Rogmans et al., 2010b]** Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Immersive GPU-Driven Biological Adaptive Stereoscopic Rendering. In *Proceedings of the 2010 3D Stereo Media Conference (3DSM '10)*, Liège, Belgium, December 2010b

- Won the *Best Poster* award.

**[Dumont et al., 2011]** Maarten Dumont, Sammy Rogmans, Steven Maesen, Karel Frederix, Johannes Taelman, and Philippe Bekaert. A Spatial Immersive Office Environment for Computer-Supported Collaborative Work: Moving Towards the Office Of The Future. In *Proceedings of the 8th International Conference on Signal Processing and Multimedia Applications (SIGMAP '11)*, pages 212–216, Seville, Spain, July 2011

**[Goorts et al., 2013b]** Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Optimization of Free Viewpoint Interpolation by Applying Adaptive Depth Plane Distributions in Plane Sweeping. In *Proceedings of the 10th International Conference on Signal Processing and Multimedia Applications (SIGMAP '13)*, pages 7–15, Reykjavík, Iceland, July 2013b

- Won the *Best Student Paper* award.

**[Dumont et al., 2014a]** Maarten Dumont, Patrik Goorts, and Gauthier Lafruit. Plane Sweeping in Eye-Gaze Corrected, Teleimmersive 3D Videoconferencing. In Branislav Kisačanin and Margrit Gelautz, editors, *Advances in Embedded Computer Vision*, volume 2 of *Advances in Computer Vision and Pattern Recognition*, pages 135–161. Springer International Publishing, November 2014a

**[Dumont et al., 2014b]** Maarten Dumont, Patrik Goorts, Steven Maesen, Philippe Bekaert, and Gauthier Lafruit. Real-Time Local Stereo Matching using Edge Sensitive Adaptive Windows. In *Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications (SIGMAP '14)*, pages 117–126, Vienna, Austria, August 2014b

**[Dumont et al., 2014c]** Maarten Dumont, Patrik Goorts, Steven Maesen, Donald Degraen, Philippe Bekaert, and Gauthier Lafruit. Iterative Refinement for Real-time Local Stereo Matching. In *Proceedings of the 4th International Conference on 3D Imaging (IC3D '14)*, 3D Stereo Media (3DSM), pages 1–8, Liége, Belgium, December 2014c

**[Goorts et al., 2014b]** Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Optimal Distribution of Computational Power in Free Viewpoint Interpolation by Depth Hypothesis Density Adaptation in Plane Sweeping. In Mohammad S. Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, volume 456 of *Communications in Computer and Information Science (CCIS)*, pages 343–358. Springer Berlin Heidelberg, September 2014b

**[Dumont et al., 2015]** Maarten Dumont, Patrik Goorts, Steven Maesen, Gauthier Lafruit, and Philippe Bekaert. Real-Time Edge-Sensitive Local Stereo Matching with Iterative Disparity Refinement. In *Communications in Computer and Information Science (CCIS)*. Springer, 2015

- Accepted and awaiting publication.

## Semi-Related Scientific Publications

**[Goorts et al., 2012a]** Patrik Goorts, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. An End-To-End System for Free Viewpoint Video for Smooth Camera Transitions. In *Proceedings of the 2nd International Conference on 3D Imaging (IC3D '12)*, 3D Stereo Media (3DSM), pages 1–7, Liège, Belgium, December 2012a

- Won the *Best Paper* award.

**[Goorts et al., 2013a]** Patrik Goorts, Cosmin Ancuti, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Real-Time Video-Based View Interpolation of Soccer Events using Depth-Selective Plane Sweeping. In *Proceedings of the 8th International Conference on Computer Vision Theory and Applications (VISAPP '13)*, pages 131–137, Barcelona, Spain, February 2013a

**[Goorts et al., 2014a]** Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Free Viewpoint Video for Soccer using Histogram-Based Validity Maps in Plane Sweeping. In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications (VISAPP '14)*, pages 378–386, Lisbon, Portugal, January 2014a

**[Goorts et al., 2014c]** Patrik Goorts, Steven Maesen, Yunjun Liu, Maarten Dumont, Philippe Bekaert, and Gauthier Lafruit. Self-Calibration of Large Scale Camera Networks. In *Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications (SIGMAP '14)*, pages 107–116, Vienna, Austria, August 2014c

## Related Student Theses

**[Brouwers et al., 2007]** Stijn Brouwers, Maarten Dumont, Yannick Francken, and Philippe Bekaert. Multi-View Depth Estimation. Master's Thesis, Hasselt University, Hasselt, Belgium, 2007

**[Vanloffeld et al., 2008]** Rembert Vanloffeld, Maarten Dumont, Yannick Francken, and Philippe Bekaert. Real-Time Detectie en Tracking voor Sportanalyse met behulp van CUDA. Master's Thesis, Hasselt University, Hasselt, Belgium, 2008

# Bibliography

[Adelson and Bergen, 1991] Edward H. Adelson and James R. Bergen. The Plenoptic Function and the Elements of Early Vision. *Computational Models of Visual Processing*, 1 (2):3–20, October 1991.

[Ambrosch et al., 2009] Kristian Ambrosch, Martin Humenberger, Wilfried Kubinger, and Andreas Steininger. SAD-Based Stereo Matching Using FPGAs. In Branislav Kisačanin, Shuvra S. Bhattacharyya, and Sek Chai, editors, *Embedded Computer Vision*, volume II of *Advances in Pattern Recognition*, pages 121–138. Springer London, 2009.

[Ansar et al., 2004] A. Ansar, A. Castano, and L. Matthies. Enhanced Real-Time Stereo using Bilateral Filtering. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '04)*, pages 455–462, Thessaloniki, Greece, September 2004.

[Aoki et al., 1999] Terumasa Aoki, W. Kustarto, Nobuki Sakamoto, N. Suzuki, K. Saburi, and Hiroshi Yasuda. MONJUnoCHIE System: Videoconference System with Eye Contact for Decision Making. In *Proceedings of the 2nd International Workshop on Advanced Image Technology (IWAIT '99)*, Taiwan, January 1999.

[ARC Science Simulations, 2003] ARC Science Simulations. OmniGlobe: A Self-Contained Spherical Display System. `http://arcscience.com`, 2003. ACM SIGGRAPH Emerging Technologies.

[Argyle, 1988] Michael Argyle. *Bodily Communication*. Routledge, 2nd edition, March 1988. ISBN 978-041-50511-4-9.

[Argyle and Cook, 1976] Michael Argyle and Mark Cook. *Gaze and Mutual Gaze*. Cambridge University Press, January 1976. ISBN 978-052-12086-5-9.

[Arun et al., 1987] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting Of 2 3-D Point Sets. *IEEE Transactions On Pattern Analysis And Machine Intelligence (TPAMI)*, 9(5):698–700, September 1987.

[Asanovic et al., 2009] Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Wawrzynek, David Wessel, and Katherine Yelick. A View of the Parallel Computing Landscape. *Communications of the ACM (CACM)*, 52(10):56–67, October 2009.

[Astrachan, 2003] Owen Astrachan. Bubble Sort: An Archaeological Algorithmic Analysis. *ACM SIGCSE Bulletin*, 35(1):1–5, January 2003.

[Avidan and Shashua, 1997] Shai Avidan and Amnon Shashua. Novel View Synthesis in Tensor Space. In *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 1034–1040, San Juan, Puerto Rico, June 1997.

[Baker et al., 2002] H. Harlyn Baker, Nina T. Bhatti, Donald Tanguay, Irwin Sobel, Dan Gelb, Michael E. Goss, W. Bruce Culbertson, and Thomas Malzbender. The Coliseum Immersive Teleconferencing System. In *Proceedings of the International Workshop on Immersive Telepresence*, Juan-les-Pins, France, December 2002.

[Baker et al., 2005] H. Harlyn Baker, Nina Bhatti, Donald Tanguay, Irwin Sobel, Dan Gelb, Michael E. Goss, W. Bruce Culbertson, and Thomas Malzbender. Understanding Performance in Coliseum, An Immersive Videoconferencing System. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 1(2):190–210, May 2005.

[Baker and Aloimonos, 2003] Patrick T. Baker and Yiannis Aloimonos. Calibration of a Multicamera Network. In *Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '03)*, volume 7, page 72, Madison, WI, USA, June 2003.

[Banz et al., 2010] Christian Banz, Sebastian Hesselbarth, Holger Flatt, Holger Blume, and Peter Pirsch. Real-Time Stereo Vision System using Semi-Global Matching Disparity Estimation: Architecture and FPGA-Implementation. In *Proceedings of the 10th International Conference on Embedded Computer Systems (SAMOS '10)*, pages 93–101, Samos, Greece, July 2010.

[Barco NV] Barco NV. I-Space Cave Display. `http://www.barco.com/en/Products-Solutions/Visual-display-systems/`. Multi-walled stereoscopic environment.

[Bay et al., 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3): 346–359, June 2008.

[Bekaert, 2008–2012] Philippe Bekaert. Immersive Display for Home Environments. 2020 3D Media Project Deliverable 6.6, 2008–2012. Seventh Framework Programme ICT-FP7-215475.

[Bergasa et al., 2000] Luis M. Bergasa, Manuel Mazo, Alfredo Gardel, Miguel A. Sotelo, and Luciano Boquete. Unsupervised and Adaptive Gaussian Skin-Color Model. *Image and Vision Computing*, 18(12):987–1003, September 2000.

[Bimber and Raskar, 2005] Oliver Bimber and Ramesh Raskar. *Spatial Augmented Reality: Merging Real and Virtual Worlds*, volume 6. A. K. Peters, Ltd., August 2005. ISBN 978-156-88123-0-4.

[Bleyer and Gelautz, 2005] Michael Bleyer and Margrit Gelautz. A Layered Stereo Matching Algorithm using Image Segmentation and Global Visibility Constraints. *Journal of Photogrammetry and Remote Sensing*, 59(3):128–150, May 2005.

[Blinn and Newell, 1976] James F. Blinn and Martin E. Newell. Texture and Reflection in Computer Generated Images. *Communications of the ACM (CACM)*, 19(10):542–547, October 1976.

[Blythe, 2006] David Blythe. The Direct3D 10 System. *ACM Transactions on Graphics (TOG)*, 25(3):724–734, July 2006.

[Bobick and Intille, 1999] Aaron F. Bobick and Stephen S. Intille. Large Occlusion Stereo. *International Journal of Computer Vision (IJCV)*, 33(3):181–200, September 1999.

[Bogomjakov et al., 2006] Alexander Bogomjakov, Craig Gotsman, and Marcus Magnor. Free-Viewpoint Video from Depth Cameras. In *Proceedings of the 11th International Workshop on Vision, Modeling, and Visualization (VMV '06)*, pages 89–96, Aachen, Germany, November 2006.

[Boyd, 2008] Chas Boyd. The DirectX 11 Compute Shader. In *Proceedings of the 35th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '08)*, volume 25, Los Angeles, CA, USA, August 2008.

[Boykov and Kolmogorov, 2004] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26 (9):1124–1137, September 2004.

[Boykov et al., 1998] Yuri Boykov, Olga Veksler, and Ramin Zabih. A Variable Window Approach to Early Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(12):1283–1294, December 1998.

[Brouwers et al., 2007] Stijn Brouwers, Maarten Dumont, Yannick Francken, and Philippe Bekaert. Multi-View Depth Estimation. Master's Thesis, Hasselt University, Hasselt, Belgium, 2007.

[Brown, 1966] Duane C. Brown. Decentering Distortion of Lenses. *Photometric Engineering*, 32(3):444–462, May 1966.

[Buehler et al., 2001] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured Lumigraph Rendering. In *Proceedings of the 28th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pages 425–432, Los Angeles, CA , USA, August 2001.

[Carranza et al., 2003] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-Viewpoint Video of Human Actors. *ACM Transactions on Graphics (TOG)*, 22(3):569–577, July 2003.

[Chai et al., 2000] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic Sampling. In *Proceedings of the 27th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pages 307–318, New Orleans, LA, USA, July 2000.

[Chang et al., 1999] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. LDI Tree: A Hierarchical Representation for Image-Based Rendering. In *Proceedings of the 26th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pages 291–298, Los Angeles, CA, USA, August 1999.

[Chen et al., 2007] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-Time Edge-Aware Image Processing with the Bilateral Grid. *ACM Transactions on Graphics (TOG)*, 26 (3):103, July 2007.

[Chen, 1995] Shenchang E. Chen. QuickTime VR: An Image-Based Approach to Virtual Environment Navigation. In *Proceedings of the 22nd ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 29–38, Los Angeles, CA, USA, August 1995.

[Chen and Williams, 1993] Shenchang E. Chen and Lance Williams. View Interpolation for Image Synthesis. In *Proceedings of the 20th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*, pages 279–288, Anaheim, CA, USA, August 1993.

[Chen et al., 2000] Wei-Chao Chen, Herman Towles, Lars Nyland, Greg Welch, and Henry Fuchs. Toward a Compelling Sensation of Telepresence: Demonstrating a Portal to a Distant (Static) Office. In *Proceedings of the 11th IEEE Conference on Visualization (VIS '00)*, pages 327–333, Salt Lake City, UT, USA, October 2000.

[Chien et al., 2003] Shao-Yi Chien, Shu-Han Yu, Li-Fu Ding, Yun-Nien Huang, and Liang-Gee Chen. Efficient Stereo Video Coding System for Immersive Teleconference with Two-Stage Hybrid Disparity Estimation Algorithm. In *Proceedings of the 10th IEEE*

*International Conference on Image Processing (ICIP '03)*, volume 1, pages I–749–52, Barcelona, Spain, September 2003.

[Childers et al., 2000] Lisa Childers, Terry Disz, Robert Olson, Michael E. Papka, Rick Stevens, and Tushar Udeshi. Access Grid: Immersive Group-To-Group Collaborative Visualization. In *Proceedings of the 4th International Immersive Projection Technology Workshop (IPT '00)*, pages 62–82, Ames, IA, US, June 2000.

[Cisco Systems, 2009] Cisco Systems. Telepresence Solutions. `http://www.cisco.com/en/US/netsol/ns669/networking_solutions_solution_segment_home.html`, 2009.

[Cohen et al., 2000] Michael F. Cohen, R. Alex Colburn, and Steven M. Drucker. The Role of Eye Gaze in Avatar Mediated Conversational Interfaces. Technical Report MSR-TR-2000-81, Microsoft Research, Redmond, WA, USA, July 2000.

[Collins, 1996] Robert T. Collins. A Space-Sweep Approach to True Multi-Image Matching. In *Proceedings of the 1996 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 358–363, San Francisco, CA, USA, June 1996.

[Comaniciu and Meer, 2002] Dorin Comaniciu and Peter Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619, May 2002.

[Cornells and Van Gool, 2005] Nico Cornells and Luc Van Gool. Real-Time Connectivity Constrained Depth Map Computation using Programmable Graphics Hardware. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, volume 1, pages 1099–1104, San Diego, CA, USA, June 2005.

[Criminisi et al., 2003] Antonio Criminisi, Jamie Shotton, Andrew Blake, and Philip H. S. Torr. Gaze Manipulation for One-To-One Teleconferencing. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, pages 191–198, Nice, France, October 2003.

[Crow, 1984] Franklin C. Crow. Summed-Area Tables for Texture Mapping. In *Proceedings of the 11th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '84)*, pages 207–212, Minneapolis, MN, USA, July 1984.

[Cruz-Neira et al., 1993] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Proceedings of the 20th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*, pages 135–142, Anaheim, CA, USA, August 1993.

[CUO] CUO. Centre for User Experience Research. `http://soc.kuleuven.be/com/mediac/cuo/`.

[Cuypers et al., 2008] Tom Cuypers, Jan Schneider, Johannes Taelman, Kris Luyten, and Philippe Bekaert. Eunomia: Toward a Framework for Multi-Touch Information Displays in Public Spaces. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction (BCS-HCI '08)*, pages 31–34, Liverpool, UK, September 2008.

[Cuypers et al., 2009] Tom Cuypers, Karel Frederix, Chris Raymaekers, and Philippe Bekaert. A Framework for Networked Interactive Surfaces. In *Proceedings of the 2nd Workshop on Software Engineering and Architecture for Realtime Interactive Systems (SEARISIEEE VR '09)*, Lafayette, LA, USA, March 2009.

[Darabiha et al., 2006] Ahmad Darabiha, W. James MacLean, and Jonathan Rose. Reconfigurable Hardware Implementation of a Phase-Correlation Stereo Algorithm. *Machine Vision and Applications (MVA)*, 17(2):116–132, May 2006.

[Davis et al., 2005] James Davis, Diego Nehab, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Spacetime Stereo: A Unifying Framework for Depth From Triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27 (2):296–302, February 2005.

[Debevec, 1998] Paul Debevec. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. In *Proceedings of the 25th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 189–198, Orlando, FL, USA, July 1998.

[Debevec, 2002] Paul Debevec. Image-Based Lighting. *IEEE Computer Graphics and Applications*, 22(2):26–34, March 2002.

[Debevec et al., 1998] Paul Debevec, Yizhou Yu, and George Boshokov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *Proceedings of the 9th Eurographics Workshop on Rendering Techniques (EGWR '98)*, pages 105–116, Vienna, Austria, June 1998.

[Debevec et al., 1996] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry and Image-based Approach. In *Proceedings of the 23rd ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 11–20, New Orleans, LA, USA, August 1996.

[Deng et al., 2007] Yi Deng, Qiong Yang, Xueyin Lin, and Xiaoou Tang. Stereo Correspondence with Occlusion Handling in a Symmetric Patch-Based Graph-Cuts Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(6): 1068–1079, June 2007.

[Devernay and Faugeras, 2001] Frédréric Devernay and Olivier Faugeras. Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Environments. *Machine Vision and Applications (MVA)*, 13(1):14–24, August 2001.

[Di Fiore et al., 2008] Fabian Di Fiore, Peter Quax, Cedric Vanaken, Wim Lamotte, and Frank Van Reeth. Conveying Emotions Through Facially Animated Avatars in Networked Virtual Environments. In Arjan Egges, Arno Kamphuis, and Mark Overmars, editors, *Motion in Games*, volume 5277 of *Lecture Notes in Computer Science (LNCS)*, pages 222–233. Springer Berlin Heidelberg, June 2008.

[Dietz and Leigh, 2001] Paul Dietz and Darren Leigh. DiamondTouch: A Multi-User Touch Technology. In *Proceedings of the 14th ACM Symposium on User Interface Software and Technology (UIST '01)*, pages 219–226, Orlando, FL, USA, November 2001.

[Divorra et al., 2010] Òscar Divorra, Jaume Civit, Fei Zuo, Harm Belt, Ingo Feldmann, Oliver Schreer, Einat Yellin, Wijnand IJsselsteijn, Rob van Eijk, David Espinola, Pierre Hagendof, Wolfgang Waizenegger, and Ralph Braspenning. Towards 3D-Aware Telepresence: Working on Technologies Behind the Scene. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW '10)*, pages 6–10, Savannah, GA, USA, February 2010.

[Dumont et al., 2008] Maarten Dumont, Steven Maesen, Sammy Rogmans, and Philippe Bekaert. A Prototype for Practical Eye-Gaze Corrected Video Chat on Graphics Hardware. In *Proceedings of the 5th International Conference on Signal Processing and Multimedia Applications (SIGMAP '08)*, pages 236–243, Porto, Portugal, July 2008.

[Dumont et al., 2009a] Maarten Dumont, Sammy Rogmans, Gauthier Lafruit, and Philippe Bekaert. Immersive Teleconferencing with Natural 3D Stereoscopic Eye Contact using GPU Computing. In *Proceedings of the 2009 3D Stereo Media Conference (3DSM '09)*, Liège, Belgium, December 2009a.

[Dumont et al., 2009b] Maarten Dumont, Sammy Rogmans, Steven Maesen, and Philippe Bekaert. Optimized Two-Party Video Chat with Restored Eye Contact using Graphics Hardware. In Joaquim Filipe and Mohammad S. Obaidat, editors, *e-Business and Telecommunications*, volume 48 of *Communications in Computer and Information Science (CCIS)*, pages 358–372. Springer Berlin Heidelberg, November 2009b.

[Dumont et al., 2010] Maarten Dumont, Steven Maesen, Karel Frederix, Chris Raymaekers, Philippe Bekaert, and Frank Van Reeth. Immersive Collaboration Environment. In Elisabetta Di Nitto and Ramin Yahyapour, editors, *Towards a Service-Based Internet*, volume 6481 of *Lecture Notes in Computer Science (LNCS)*, pages 187–188. Springer Berlin Heidelberg, 2010.

[Dumont et al., 2011] Maarten Dumont, Sammy Rogmans, Steven Maesen, Karel Frederix, Johannes Taelman, and Philippe Bekaert. A Spatial Immersive Office Environment for Computer-Supported Collaborative Work: Moving Towards the Office Of The Future. In *Proceedings of the 8th International Conference on Signal Processing and Multimedia Applications (SIGMAP '11)*, pages 212–216, Seville, Spain, July 2011.

[Dumont et al., 2014a] Maarten Dumont, Patrik Goorts, and Gauthier Lafruit. Plane Sweeping in Eye-Gaze Corrected, Teleimmersive 3D Videoconferencing. In Branislav Kisačanin and Margrit Gelautz, editors, *Advances in Embedded Computer Vision*, volume 2 of *Advances in Computer Vision and Pattern Recognition*, pages 135–161. Springer International Publishing, November 2014a.

[Dumont et al., 2014b] Maarten Dumont, Patrik Goorts, Steven Maesen, Philippe Bekaert, and Gauthier Lafruit. Real-Time Local Stereo Matching using Edge Sensitive Adaptive Windows. In *Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications (SIGMAP '14)*, pages 117–126, Vienna, Austria, August 2014b.

[Dumont et al., 2014c] Maarten Dumont, Patrik Goorts, Steven Maesen, Donald Degraen, Philippe Bekaert, and Gauthier Lafruit. Iterative Refinement for Real-time Local Stereo Matching. In *Proceedings of the 4th International Conference on 3D Imaging (IC3D '14)*, 3D Stereo Media (3DSM), pages 1–8, Liége, Belgium, December 2014c.

[Dumont et al., 2015] Maarten Dumont, Patrik Goorts, Steven Maesen, Gauthier Lafruit, and Philippe Bekaert. Real-Time Edge-Sensitive Local Stereo Matching with Iterative Disparity Refinement. In *Communications in Computer and Information Science (CCIS)*. Springer, 2015.

[Durand and Dorsey, 2002] Frédo Durand and Julie Dorsey. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. *ACM Transactions on Graphics (TOG)*, 21 (3):257–266, July 2002.

[Eisemann et al., 2008] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. *Computer Graphics Forum*, 27(2):409–418, April 2008.

[Eric Joris] Eric Joris. CREW. `http://www.crewonline.org`. CREW is a company that operates on the border between art and science, between performance art and new technology.

[ETRO] ETRO. VUB-Department of Electronics and Informatics. `http://www.etro.vub.ac.be`.

[Faugeras et al., 1993] Olivier Faugeras, Bernard Hotz, Hervé Mathieu, Thierry Viéville, Zhengyou Zhang, Pascal Fua, Eric Théron, Laurent Moll, Gérard Berry, and Jean Vuillemin. Real Time Correlation-Based Stereo: Algorithm, Implementations and Applications. Technical report, French Institute for Research in Computer Science and Automation (INRIA), Rocquencourt, France, 1993.

[Feldmann et al., 2003] Ingo Feldmann, Oliver Schreer, and Peter Kauff. Nonlinear Depth Scaling for Immersive Video Applications. In *Proceedings of the 4th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '03)*, pages 433–438, London, UK, April 2003.

[Feldmann et al., 2009a] Ingo Feldmann, Nicole Atzpadin, Oliver Schreer, Jose-Carlos Pujol-Acolado, José L. Landabaso, and Òscar D. Escoda. Multi-View Depth Estimation Based on Visual-Hull Enhanced Hybrid Recursive Matching for 3D Video Conference Systems. In *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP '09)*, pages 745–748, Cairo, Egypt, November 2009a.

[Feldmann et al., 2009b] Ingo Feldmann, Wolfgang Waizenegger, Nicole Atzpadin, and Oliver Schreer. Immersive Multi-User 3D Video Communication. In *Proceedings of the International Broadcasting Convention (IBC '09)*, Amsterdam, The Netherlands, September 2009b.

[Feldmann et al., 2010] Ingo Feldmann, Wolfgang Waizenegger, Nicole Atzpadin, and Oliver Schreer. Real-Time Depth Estimation for Immersive 3D Videoconferencing. In *Proceedings of the 2010 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON '10)*, pages 1–4, Tampere, Finland, June 2010.

[Fernando, 2004] Randima Fernando. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*. Addison-Wesley Professional, 1st edition, April 2004. ISBN 978-032-12283-2-1.

[Fiala, 2005] Mark Fiala. Automatic Projector Calibration using Self-Identifying Patterns. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '05)*, page 113, San Diego, CA, USA, June 2005.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM (CACM)*, 24(6):381–395, June 1981.

[Fitzgibbon and Zisserman, 1998] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic Camera Recovery for Closed or Open Image Sequences. In *Proceedings of the 5th European Conference on Computer Vision (ECCV '98)*, volume I, pages 311–326, Freiburg, Germany, June 1998.

[Forstmann et al., 2004] Sven Forstmann, Yutaka Kanou, Jun Ohya, Sven Thuering, and Alfred Schmitt. Real-Time Stereo by using Dynamic Programming. In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '04)*, pages 29–29, Washington, DC, USA, June 2004.

[Francken et al., 2007] Yannick Francken, Chris Hermans, and Philippe Bekaert. Screen-Camera Calibration using a Spherical Mirror. In *Proceedings of the 4th Canadian Conference on Computer and Robot Vision (CRV '07)*, pages 11–20, Montreal, QC, Canada, May 2007.

[Fusiello and Roberto, 1997] Andrea Fusiello and Vito Roberto. Efficient Stereo with Multiple Windowing. In *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 858–863, San Juan, Puerto Rico, June 1997.

[Gallup et al., 2007] David Gallup, Jan-Michael Frahmm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–8, Minneapolis, MN, USA, June 2007.

[Gemmell et al., 2000] Jim Gemmell, Kentaro Toyama, C. Lawrence Zitnick, Thomas Kang, and Steven Seitz. Gaze Awareness for Video-Conferencing: A Software Approach. *IEEE MultiMedia*, 7(4):26–35, October 2000.

[Gerrits and Bekaert, 2006] Mark Gerrits and Philippe Bekaert. Local Stereo Natching with Segmentation-Based Outlier Rejection. In *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV '06)*, pages 66–66, Quebec, QC, Canada, June 2006.

[Geys et al., 2004] Indra Geys, Thomas P. Koninckx, and Luc Van Gool. Fast Interpolated Cameras by Combining a GPU Based Plane Sweep with a Max-Flow Regularisation Algorithm. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '04)*, pages 534–541, Thessaloniki, Greece, September 2004.

[Gibbs et al., 1999] Simon J. Gibbs, Constantin Arapis, and Christian J. Breiteneder. TELEPORT – Towards Immersive Copresence. *Multimedia Systems*, 7(3):214–221, May 1999.

[Giger et al., 2014] Dominik Giger, Jean-Charles Bazin, Claudia Kuster, Tiberiu Popa, and Markus Gross. Gaze Correction with a Single Webcam. In *Proceedings of the 2014 IEEE International Conference on Multimedia and Expo (ICME '14)*, pages 1–6, Chengdu, China, July 2014.

[Global Imagination, 2002] Global Imagination. Magic Planet. `http://globalimagination.com`, August 2002.

[Golub and Van Loan, 1996] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, October 1996. ISBN 978-080-18541-4-9.

[Gong, 2006] Minglun Gong. Enforcing Temporal Consistency in Real-Time Stereo Estimation. In *Proceedings of the 9th European Conference on Computer Vision (ECCV '06)*, pages 564–577, Graz, Austria, May 2006.

[Gong and Yang, 2005] Minglun Gong and Ruigang Yang. Image-Gradient-Guided Real-Time Stereo on Graphics Hardware. In *Proceedings of the 5th International Conference on 3-D Digital Imaging and Modeling (3DIM '05)*, pages 548–555, Ottawa, ON, Canada, June 2005.

[Gong et al., 2007] Minglun Gong, Ruigang Yang, Liang Wang, and Mingwei Gong. A Performance Study on Different Cost Aggregation Approaches used in Real-Time Stereo Matching. *International Journal of Computer Vision (IJCV)*, 75(2):283–296, November 2007.

[Goorts, 2014] Patrik Goorts. *Real-Time, Adaptive Plane Sweeping for Free Viewpoint Navigation in Soccer Scenes*. PhD Thesis, Hasselt University, Hasselt, Belgium, June 2014.

[Goorts et al., 2009] Patrik Goorts, Sammy Rogmans, and Philippe Bekaert. Optimal Data Distribution for Versatile Finite Impulse Response Filtering on Next-Generation Graphics Hardware using CUDA. In *Proceedings of the 15th IEEE International Conference on Parallel and Distributed Systems (ICPADS '09)*, pages 300–307, Shenzhen, China, December 2009.

[Goorts et al., 2010] Patrik Goorts, Sammy Rogmans, Steven Vanden Eynde, and Philippe Bekaert. Practical Examples of GPU Computing Optimization Principles. In *Proceedings of the 7th International Conference on Signal Processing and Multimedia Applications (SIGMAP '10)*, pages 46–49, Athens, Greece, July 2010.

[Goorts et al., 2012a] Patrik Goorts, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. An End-To-End System for Free Viewpoint Video for Smooth Camera Transitions. In *Proceedings of the 2nd International Conference on 3D Imaging (IC3D '12)*, 3D Stereo Media (3DSM), pages 1–7, Liège, Belgium, December 2012a.

[Goorts et al., 2012b] Patrik Goorts, Sammy Rogmans, and Philippe Bekaert. Raw Camera Image Demosaicing using Finite Impulse Response Filtering on Commodity GPU Hardware using CUDA. In *Proceedings of the 9th International Conference on Signal Processing and Multimedia Applications (SIGMAP '12)*, pages 96–101, Rome, Italy, July 2012b.

[Goorts et al., 2013a] Patrik Goorts, Cosmin Ancuti, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Real-Time Video-Based View Interpolation of Soccer Events using Depth-Selective Plane Sweeping. In *Proceedings of the 8th International Conference on Computer Vision Theory and Applications (VISAPP '13)*, pages 131–137, Barcelona, Spain, February 2013a.

[Goorts et al., 2013b] Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Optimization of Free Viewpoint Interpolation by Applying Adaptive Depth Plane Distributions in Plane Sweeping. In *Proceedings of the 10th International Conference on Signal Processing and Multimedia Applications (SIGMAP '13)*, pages 7–15, Reykjavík, Iceland, July 2013b.

[Goorts et al., 2014a] Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Free Viewpoint Video for Soccer using Histogram-Based Validity Maps in Plane Sweeping. In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications (VISAPP '14)*, pages 378–386, Lisbon, Portugal, January 2014a.

[Goorts et al., 2014b] Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Optimal Distribution of Computational Power in Free Viewpoint Interpolation by Depth Hypothesis Density Adaptation in Plane Sweeping. In Mohammad S. Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, volume 456 of *Communications in Computer and Information Science (CCIS)*, pages 343–358. Springer Berlin Heidelberg, September 2014b.

[Goorts et al., 2014c] Patrik Goorts, Steven Maesen, Yunjun Liu, Maarten Dumont, Philippe Bekaert, and Gauthier Lafruit. Self-Calibration of Large Scale Camera Networks. In *Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications (SIGMAP '14)*, pages 107–116, Vienna, Austria, August 2014c.

[Gortler et al., 1996] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *Proceedings of the 23rd ACM International Conference on*

*Computer Graphics and Interactive Techniques (SIGGRAPH ’96)*, pages 43–54, New Orleans, LA, USA, August 1996.

[Gramkow, 2001] Claus Gramkow. On Averaging Rotations. *International Journal of Computer Vision (IJCV)*, 42(1–2):7–16, April 2001.

[Green and Whites, 2000] Mark Green and Lloyd Whites. The Cave-let: A Low-Cost Projective Immersive Display. *Journal of Telemedicine and Telecare*, 6(2):24–26, June 2000.

[Green, 2005] Simon Green. Image Processing Tricks in OpenGL. In *Proceedings of the Game Developers Conference (GDC ’05)*, page 2, San Francisco, CA, USA, March 2005.

[Griesser and Van Gool, 2006] Andreas Griesser and Luc Van Gool. Automatic Interactive Calibration of Multi-Projector-Camera Systems. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW ’06)*, page 8, New York, NY, USA, June 2006.

[Gross et al., 2003] Markus Gross, Stephan Würmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, Andrew Vande Moere, and Oliver Staadt. Blue-C: A Spatially Immersive Display and 3D Video Portal for Telepresence. *ACM Transactions on Graphics (TOG)*, 22(3):819–827, July 2003.

[Guo et al., 2005] Xun Guo, Wen Gao, and Debin Zhao. Motion Vector Prediction in Multiview Video Coding. In *Proceedings of the 12th IEEE International Conference on Image Processing (ICIP ’05)*, volume 2, pages II–337, Genova, Italy, September 2005.

[Han, 2005] Jefferson Y. Han. Low-Cost Multi-Touch Sensing Through Frustrated Total Internal Reflection. In *Proceedings of the 18th ACM Symposium on User Interface Software and Technology (UIST ’05)*, pages 115–118, Seattle, WA, USA, October 2005.

[Hartley, 1997] Richard I. Hartley. In Defense of the Eight-Point Algorithm. *IEEE Transactions On Pattern Analysis And Machine Intelligence (TPAMI)*, 19(6):580–593, June 1997.

[Hartley and Zisserman, 2004] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, March 2004. ISBN 052-1-54051-8.

[Harville et al., 2006] Michael Harville, Bruce Culbertson, Irwin Sobel, Dan Gelb, Andrew Fitzhugh, and Donald Tanguay. Practical Methods for Geometric and Photometric

Correction of Tiled Projector. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '06)*, page 5, New York, NY, USA, June 2006.

[Hayashi and Saito, 2006] Kunihiko Hayashi and Hideo Saito. Synthesizing Free-Viewpoint Images from Multiple View Videos in Soccer Stadium. In *Proceedings of the 3rd International Conference on Computer Graphics, Imaging and Visualisation (CGIV '06)*, pages 220–225, Sydney, NSW, Australia, July 2006.

[He et al., 2010] Kaiming He, Jian Sun, and Xiaoou Tang. Guided Image Filtering. In *Proceedings of the 11th European Conference on Computer Vision (ECCV '10)*, pages 1–14, Heraklion, Greece, September 2010.

[Hebert et al., 1995] Martial Hebert, Cyril Zeller, Olivier Faugeras, and Luc Robert. Applications of Non-Metric Vision to Some Visually Guided Robotics Tasks. Technical Report 2584, French Institute for Research in Computer Science and Automation (INRIA), Rocquencourt, France, June 1995.

[Held et al., 2010] Robert T. Held, Emily A. Cooper, James F. O'Brien, and Martin S. Banks. Using Blur to Affect Perceived Distance and Size. *ACM Transactions on Graphics (TOG)*, 29(2):19:1–19:16, April 2010.

[Hermans, 2011] Chris Hermans. *Mobile Structured Light: Reconstruction using a Signal Processing Approach*. PhD Thesis, Hasselt University, Hasselt, Belgium, April 2011.

[Hermans et al., 2006] Chris Hermans, Maarten Dumont, Philippe Bekaert, and Eric Joris. JanusLights: A Camera-Projection System for Telematic Omni-Presence with Correct Eye Gaze. In *Proceedings of the 3rd IEEE International Workshop on Projector-Camera Systems (PROCAMS '06)*, pages 5–6, New York, NY, USA, June 2006.

[Hermans et al., 2007a] Chris Hermans, Maarten Dumont, and Philippe Bekaert. JanusLights: A Camera-Projection System for Telematic Omni-Presence with Correct Eye Gaze. In *Proceedings of the IX Symposium on Virtual and Augmented Reality (SVR '07)*, Petrópolis, Rio de Janeiro, Brazil, May 2007a.

[Hermans et al., 2007b] Chris Hermans, Maarten Dumont, and Philippe Bekaert. Extrinsic Recalibration in Camera Networks. In *Proceedings of the 4th Canadian Conference on Computer and Robot Vision (CRV '07)*, pages 3–10, Montreal, QC, Canada, May 2007b.

[Hermans et al., 2008] Chris Hermans, Cedric Vanaken, Tom Mertens, Frank Van Reeth, and Philippe Bekaert. Augmented Panoramic Video. *Computer Graphics Forum*, 27(2): 281–290, April 2008.

[Hirakawa and Parks, 2005] Keigo Hirakawa and Thomas W. Parks. Adaptive Homogeneity-Directed Demosaicing Algorithm. *IEEE Transactions on Image Processing*, 14(3): 360–369, March 2005.

[Hirschmüller et al., 2002] Heiko Hirschmüller, Peter R. Innocent, and Jon Garibaldi. Real-Time Correlation-Based Stereo Vision with Reduced Border Errors. *International Journal of Computer Vision (IJCV)*, 47(1–3):229–246, April 2002.

[Holovis, 2013] Holovis. MotionDome. `http://www.holovis.com`, 2013. The next generation of dark ride and interactive gaming solutions.

[Holzmann and Strobl, 2005] Georg Holzmann and Gerda Strobl. Adaptive: Normalized Least Mean Square (NLMS). `http://grh.mur.at/software/adaptive.html`, 2005. A Pure Data external library for adaptive systems and filters.

[Horn, 1987] Berthold K. P. Horn. Closed-Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, April 1987.

[Hosni et al., 2009] Asmaa Hosni, Michael Bleyer, Margrit Gelautz, and Christoph Rhemann. Local Stereo Matching using Geodesic Support Weights. In *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP '09)*, pages 2093–2096, Cairo, Egypt, November 2009.

[Hosni et al., 2013] Asmaa Hosni, Michael Bleyer, and Margrit Gelautz. Secrets of Adaptive Support Weight Techniques for Local Stereo Matching. *Computer Vision and Image Understanding*, 117(6):620–632, June 2013.

[Hsu et al., 2002] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain. Face Detection in Color Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):696–706, May 2002.

[IBBT ISBO VIN project, 2005–2008] IBBT ISBO VIN project. Virtual Individual Networks. `http://www.iminds.be`, 2005–2008. Interdisciplinary Institute for Broadband Technology.

[Isgro et al., 2004] Francesco Isgro, Emanuele Trucco, Peter Kauff, and Oliver Schreer. Three-Dimensional Image Processing in the Future of Immersive Media. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 14(3):288–303, March 2004.

[Jacobs, 1997] David Jacobs. Linear Fitting with Missing Data: Applications to Structure-From-Motion and to Characterizing Intensity Images. In *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 206–212, San Juan, Puerto Rico, June 1997.

[Jerald and Daily, 2002] Jason Jerald and Mike Daily. Eye Gaze Correction for Videoconferencing. In *Proceedings of the 2002 Symposium on Eye Tracking Research & Applications (ETRA '02)*, pages 77–81, New Orleans, LA, USA, March 2002.

[Jones et al., 2013] Brett R. Jones, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. IllumiRoom: Peripheral Projected Illusions for Interactive Experiences. In *Proceedings of the 31th SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, pages 869–878, Paris, France, April 2013.

[Jorissen et al., 2014] Lode Jorissen, Patrik Goorts, Bram Bex, Nick Michiels, Sammy Rogmans, Philippe Bekaert, and Gauthier Lafruit. A Qualitative Comparison of MPEG View Synthesis and Light Field Rendering. In *Proceedings of the 2014 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON '14)*, pages 1–4, Budapest, Hungary, July 2014.

[Juarez et al., 2010] Alex Juarez, Willem Schonenberg, and Christoph Bartneck. Implementing a Low-Cost CAVE System using the CryEngine2. *Entertainment Computing*, 1 (3–4):157–164, December 2010.

[Kanade and Okutomi, 1994] Takeo Kanade and Masatoshi Okutomi. A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(9):920–932, September 1994.

[Kang et al., 2001] Sing Bing Kang, Richard Szeliski, and Jinxiang Chai. Handling Occlusions in Dense Multi-View Stereo. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '01)*, volume 1, pages I–103, Kauai, HI, USA, December 2001.

[Kang et al., 2004] Sing Bing Kang, Richard Szeliski, and Matthew Uyttendaele. Seamless Stitching using Multi-Perspective Plane Sweep. Technical Report MSR-TR-2004-48, Microsoft Research, Redmond, WA, USA, 2004.

[Kauff and Schreer, 2002] Peter Kauff and Oliver Schreer. An Immersive 3D Video-Conferencing System using Shared Virtual Team User Environments. In *Proceedings of the 4th International Conference on Collaborative Virtual Environments (CVE '02)*, pages 105–112, Bonn, Germany, September 2002.

[Kim et al., 2005] Jae C. Kim, Kyoung M. Lee, Byoung T. Choi, and Sang U. Lee. A Dense Stereo Matching using Two-Pass Dynamic Programming with Generalized Ground Control Points. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, volume 2, pages 1075–1082, San Diego, CA, USA, June 2005.

[Klaus et al., 2006] Andreas Klaus, Mario Sormann, and Konrad Karner. Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, volume 3, pages 15–18, Hong Kong, August 2006.

[Kolmogorov and Zabih, 2001] Vladimir Kolmogorov and Ramin Zabih. Computing Visual Correspondence with Occlusions using Graph Cuts. In *Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV '01)*, volume 2, pages 508–515, Vancouver, BC, Canada, July 2001.

[Koppel et al., 2012] Martin Koppel, Xi Wang, Dimitar Doshkov, Thomas Wiegand, and Patrick Ndjiki-Nya. Depth Image-Based Rendering with Spatio-Temporally Consistent Texture Synthesis for 3-D Video With Global Motion. In *Proceedings of the 19th IEEE International Conference on Image Processing (ICIP '12)*, pages 2713–2716, Orlando, FL, USA, September 2012.

[Kuster et al., 2012] Claudia Kuster, Tiberiu Popa, Jean-Charles Bazin, Craig Gotsman, and Markus Gross. Gaze Correction for Home Video Conferencing. *ACM Transactions on Graphics (TOG)*, 31(6):174:1–174:6, November 2012.

[Laurentini, 1994] Aldo Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(2):150–162, February 1994.

[Lazaros et al., 2008] Nalpantidis Lazaros, Georgios Christou Sirakoulis, and Antonios Gasteratos. Review of Stereo Vision Algorithms: From Software to Hardware. *International Journal of Optomechatronics*, 2(4):435–462, 2008.

[Lei and Hendriks, 2002] Bangjun Lei and Emile A. Hendriks. Real-Time Multi-Step View Reconstruction for a Virtual Teleconference System. *EURASIP Journal on Advances in Signal Processing (JASP)*, 2002(10):1067–1087, October 2002.

[Levoy and Hanrahan, 1996] Marc Levoy and Pat Hanrahan. Light Field Rendering. In *Proceedings of the 23rd ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 31–42, New Orleans, LA, USA, August 1996.

[Li and Chen, 1999] Kai Li and Yuqun Chen. Optical Blending for Multi-Projector Display Wall Systems. In *IEEE 12th Annual Meeting of Lasers and Electro-Optics Society (LEOS '99)*, volume 1, pages 281–282, San Francisco, CA, USA, November 1999.

[Lindholm et al., 2008] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro*, 28(2): 39–55, March 2008.

[Lowe, 2004] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, November 2004.

[Lu et al., 2007] Jiangbo Lu, Sammy Rogmans, Gauthier Lafruit, and Francky Catthoor. High-Speed Stream-Centric Dense Stereo and View Synthesis on Graphics Hardware. In *Proceedings of the 9th IEEE Workshop on Multimedia Signal Processing (MMSP '07)*, pages 243–246, Chania, Greece, October 2007.

[Lu et al., 2009] Jiangbo Lu, Sammy Rogmans, Gauthier Lafruit, and Francky Catthoor. Stream-Centric Stereo Matching and View Synthesis: A High-Speed Image-Based Rendering Paradigm on GPUs. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 19(11):1598–1611, November 2009.

[Lucas and Kanade, 1981] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, volume 2, pages 674–679, Vancouver, BC, Canada, 1981.

[Luebke and Humphreys, 2007] David Luebke and Greg Humphreys. How GPUs Work. *IEEE Computer*, 40(2):126–130, February 2007.

[Maesen, 2016] Steven Maesen. *Vision Based Tracking with Applications in Virtual Reality and Free Viewpoint Video*. PhD Thesis, Hasselt University, Hasselt, Belgium, 2016. Tentative title, to be defended.

[Maesen et al., 2013] Steven Maesen, Patrik Goorts, and Philippe Bekaert. Scalable Optical Tracking for Navigating Large Virtual Environments using Spatially Encoded Markers. In *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology (VRST '13)*, pages 101–110, Singapore, October 2013.

[Magnor, 2005] Marcus A. Magnor. *Video-Based Rendering*. A. K. Peters, Ltd., August 2005. ISBN 978-156-88124-4-1.

[Maimone et al., 2012] Andrew Maimone, Jonathan Bidwell, Kun Peng, and Henry Fuchs. Augmented Reality: Enhanced Personal Autostereoscopic Telepresence System using Commodity Depth Cameras. *Computers and Graphics*, 36(7):791–807, November 2012.

[Malvar et al., 2004] Henrique S. Malvar, Li-Wei He, and Ross Cutler. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. In *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, pages iii–485, Montreal, QC, Canada, May 2004.

[Mantzel et al., 2004] William E. Mantzel, Hyeokho Choi, and Richard G. Baraniuk. Distributed Camera Network Localization. In *Conference Record of the 38th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1381–1386, Pacific Grove, CA, USA, November 2004.

[Mark et al., 2003] William R. Mark, R. Steven Glanville, Kurt Akeley, and Mark J. Kilgard. Cg: A System for Programming Graphics Hardware in a C-Like Language. *ACM Transactions on Graphics (TOG)*, 22(3):896–907, July 2003.

[Martinec and Pajdla, 2002] Daniel Martinec and Tomáš Pajdla. Structure from Many Perspective Images with Occlusions. In *Proceedings of the 7th European Conference on Computer Vision (ECCV '02)*, volume II, pages 355–369, Copenhagen, Denmark, May 2002.

[Matas et al., 2004] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions. *Image and Vision Computing*, 22(10):761–767, September 2004.

[Matusik et al., 2000] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Leonard McMillan, and Steven Gortler. Image-Based Visual Hulls. In *Proceedings of the 27th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pages 369–374, New Orleans, LA, USA, July 2000.

[McCamy et al., 1976] Calvin S. McCamy, H. Marcus, and J. G. Davidson. A Color-Rendition Chart. *Journal of Applied Photographic Engineering*, 2(3):95–99, 1976.

[McCool, 2007] Michael D. McCool. Signal Processing and General-Purpose Computing on GPUs. *IEEE Signal Processing Magazine*, 24(3):109–114, May 2007.

[McMillan, 1997] Leonard McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD Thesis, University of North Carolina, Chapel Hill, NC, USA, 1997.

[McMillan and Bishop, 1995] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. In *Proceedings of the 22nd ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 39–46, Los Angeles, CA, USA, August 1995.

[Mechant et al., 2008] Peter Mechant, Steve Paulussen, Tim Van Lier, and Bram Lievens. Groepsgesprek Rond Videofonie in het Kader van de VIN Demonstrator Ontwikkeld door EDM. IBBT ISBO VIN Project Deliverable 4.4, 2008.

[Merrell et al., 2007] Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David Nistér, and Marc Pollefeys. Real-Time

Visibility-Based Fusion of Depth Maps. In *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV '07)*, pages 1–8, Rio de Janeiro, Brazil, October 2007.

[Michiels et al., 2014] Nick Michiels, Lode Jorissen, Jeroen Put, and Philippe Bekaert. Interactive Augmented Omnidirectional Video with Realistic Lighting. In Lucio Tommaso De Paolis and Antonio Mongelli, editors, *Augmented and Virtual Reality*, volume 8853 of *Lecture Notes in Computer Science (LNCS)*, pages 247–263. Springer International Publishing, December 2014.

[MICT] MICT. Research Group For Media & ICT. `http://www.mict.be`.

[Mikolajczyk and Schmid, 2005] Krystian Mikolajczyk and Cordelia Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(10):1615–1630, October 2005.

[Mikolajczyk et al., 2005] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision (IJCV)*, 65(1–2):43–72, November 2005.

[Min and Sohn, 2008] Dongbo Min and Kwanghoon Sohn. Cost Aggregation and Occlusion Handling with WLS in Stereo Matching. *IEEE Transactions on Image Processing*, 17 (8):1431–1442, August 2008.

[Min et al., 2011] Dongbo Min, Jiangbo Lu, and Minh N. Do. A Revisit to Cost Aggregation in Stereo Matching: How Far Can We Reduce Its Computational Redundancy? In *Proceedings of the 13th IEEE International Conference on Computer Vision (ICCV '11)*, pages 1567–1574, Barcelona, Spain, November 2011.

[Moré, 1978] Jorge J. Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. In G. A. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics (LNM)*, pages 105–116. Springer Berlin Heidelberg, 1978.

[Morgan and Krueger, 1997] David L. Morgan and Richard A. Krueger. *Planning Focus Groups (The Focus Group Kit)*, volume 1–6. SAGE Publications, 1997. ISBN 978-076-19076-0-2.

[Mori et al., 2012] Masahiro Mori, Karl F. MacDorman, and Norri Kageki. The Uncanny Valley. *IEEE Robotics & Automation Magazine*, 19(2):98–100, June 2012.

[Nalwa, 1996] Vishvjit S. Nalwa. A True Omni-Directional Viewer. Technical report, AT&T Bell Laboratories, Holmdel, NJ, USA, February 1996.

[Nashel, 2010] Andrew R. Nashel. *Rendering and Display for Multi-Viewer Tele-Immersion*. PhD Thesis, University of North Carolina, Chapel Hill, NC, USA, 2010.

[Nayar, 1997] Shree K. Nayar. Catadioptric Omnidirectional Camera. In *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 482–488, San Juan, Puerto Rico, June 1997.

[Ndjiki-Nya et al., 2011] Patrick Ndjiki-Nya, Martin Koppel, Dimitar Doshkov, Haricharan Lakshman, Philipp Merkle, Karsten Muller, and Thomas Wiegand. Depth Image-Based Rendering with Advanced Texture Synthesis for 3-D Video. *IEEE Transactions on Multimedia*, 13(3):453–465, June 2011.

[Nishihara, 1984] Herbert K. Nishihara. PRISM: A Practical Real-Time Imaging Stereo Matcher. Technical report, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, 1984.

[Novick et al., 1996] David G. Novick, Brian Hansen, and Karen Ward. Coordinating Turn-Taking with Gaze. In *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP '96)*, volume 3, pages 1888–1891, Philadelphia, PA, USA, October 1996.

[Nozick et al., 2006] Vincent Nozick, Sylvain Michelin, and Didier Arquès. Real-Time Plane-Sweep with Local Strategy. *Journal of the 14th Winter School of Computer Graphics (WSCG '06)*, 14(1–3):121–128, January 2006.

[NVIDIA Corporation, 2005] NVIDIA Corporation. SDK 9.51 Featured Code Samples: Image Histogram. `http://http.download.nvidia.com/developer/SDK/Individual_Samples/featured_samples.html#imgproc_histogram`, May 2005. This example implements an image histogram using occlusion queries.

[NVIDIA Corporation, 2007] NVIDIA Corporation. CUDA Developer Zone. `https://developer.nvidia.com/cuda-zone/`, June 2007.

[NVIDIA Corporation, 2009] NVIDIA Corporation. NVIDIA's Next Generation CUDA Compute Architecture: Fermi. Whitepaper, Santa Clara, CA, USA, 2009.

[NVIDIA Corporation, 2012] NVIDIA Corporation. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110. Whitepaper, Santa Clara, CA, USA, 2012. The Fastest, Most Efficient HPC Architecture Ever Built.

[NVIDIA Corporation, 2015] NVIDIA Corporation. CUDA C Programming Guide. `http://docs.nvidia.com/cuda/cuda-c-programming-guide/`, March 2015.

[Otto et al., 2006] Oliver Otto, Dave Roberts, and Robin Wolff. A Review on Effective Closely-Coupled Collaboration using Immersive CVE's. In *Proceedings of the 6th*

*ACM International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI '06)*, pages 145–154, Hong Kong, June 2006.

[Owens et al., 2007] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Kruger, Aaron E. Lefohn, and Timothy J. Purcell. A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, 26(1):80–113, March 2007.

[Owens et al., 2008] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. GPU Computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[Pajdla et al., 2001] Tomáš Pajdla, Tomáš Svoboda, and Vaclav Hlaváč. Epipolar Geometry of Central Panoramic Catadioptric Cameras. In Ryad Benosman and SingBing Kang, editors, *Panoramic Vision*, Monographs in Computer Science, pages 73–102. Springer-Verlag New York, 2001.

[Papadakis and Caselles, 2010] Nicolas Papadakis and Vicent Caselles. Multi-Label Depth Estimation for Graph Cuts Stereo Problems. *Journal of Mathematical Imaging and Vision (JMIV)*, 38(1):70–82, September 2010.

[Peleg et al., 2001] Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. Omnistereo: Panoramic Stereo Imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(3):279–290, March 2001.

[Point Grey] Point Grey. Innovation in Imaging. `http://www.ptgrey.com`.

[Pollefeys et al., 2008] Marc Pollefeys, David Nistér, Jan-Michael Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, Seon J. Kim, and Paul Merrell. Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision (IJCV)*, 78(2–3):143–167, July 2008.

[Polycom, 2009] Polycom. RealPresence Immersive Studio. `http://www.polycom.com/products-services/hd-telepresence-video-conferencing/realpresence-immersive.html`, 2009. Fully immersive collaboration for meetings that matter.

[Preece, 2001] Jenny Preece. Sociability and Usability in Online Communities: Determining and Measuring Success. *Behavior and Information Technology*, 20(5):347–356, 2001.

[Pritchett and Zisserman, 1998] Philip Pritchett and Andrew Zisserman. Wide Baseline Stereo Matching. In *Proceedings of the 6th IEEE International Conference on Computer Vision (ICCV '98)*, pages 754–760, Mumbai, India, January 1998.

[Puckette, 1996] Miller S. Puckette. Pure Data (Pd). `http://puredata.info`, 1996. An open source visual programming language.

[Ramanath et al., 2002] Rajeev Ramanath, Wesley E. Snyder, Griff L. Bilbro, and William A. Sander. Demosaicking Methods for Bayer Color Arrays. *Journal of Electronic Imaging*, 11(3):306–315, 2002.

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

[Raskar et al., 1999] Ramesh Raskar, Michael S. Brown, Ruigang Yang, Wei-Chao Chen, Greg Welch, Herman Towles, Brent Scales, and Henry Fuchs. Multi-Projector Displays using Camera-Based Registration. In *Proceedings of the 10th IEEE Conference on Visualization (VIS '99)*, pages 161–522, San Francisco, CA, USA, October 1999.

[Raskar et al., 2001] Ramesh Raskar, Greg Welch, Kok-Lim Low, and Deepak Bandyopadhyay. Shader Lamps: Animating Real Objects with Image-Based Illumination. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques (EGWR '01)*, pages 89–102, London, UK, June 2001.

[Raskar et al., 2003] Ramesh Raskar, Jeroen van Baar, Paul Beardsley, Thomas Willwacher, Srinivas Rao, and Clifton Forlines. iLamps: Geometrically Aware and Self-Configuring Projectors. *ACM Transactions on Graphics (TOG)*, 22(2):809–818, July 2003.

[Rhemann et al., 2011] Christoph Rhemann, Asmaa Hosni, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast Cost-Volume Filtering for Visual Correspondence and Beyond. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pages 3017–3024, Colorado Springs, CO, USA, June 2011.

[Richardt et al., 2010] Christian Richardt, Douglas Orr, Ian Davies, Antonio Criminisi, and Neil A. Dodgson. Real-Time Spatiotemporal Stereo Matching using the Dual-Cross-Bilateral Grid. In *Proceedings of the 11th European Conference on Computer Vision (ECCV '10)*, pages 510–523, Heraklion, Greece, September 2010.

[Rogmans, 2013] Sammy Rogmans. *Intelligent Visual Supercomputing on Hybrid Graphical Multiprocessor Environments*. PhD Thesis, Hasselt University, Hasselt, Belgium, May 2013.

[Rogmans et al., 2009a] Sammy Rogmans, Maarten Dumont, Tom Cuypers, Gauthier Lafruit, and Philippe Bekaert. Complexity Reduction of Real-Time Depth Scanning on Graphics Hardware. In *Proceedings of the 4th International Conference on Computer Vision Theory and Applications (VISAPP '09)*, pages 547–550, Lisbon, Portugal, February 2009a.

[Rogmans et al., 2009b] Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Migrating Real-Time Image-Based Rendering from Traditional to Next-Gen GPGPU. In *Proceedings of the 2009 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, pages 1–4, Potsdam, Germany, May 2009b.

[Rogmans et al., 2009c] Sammy Rogmans, Jiangbo Lu, Philippe Bekaert, and Gauthier Lafruit. Real-Time Stereo-Based View Synthesis Algorithms: A Unified Framework and Evaluation on Commodity GPUs. *Signal Processing: Image Communication*, 24 (1–2):49–64, January 2009c. Special Issue on Advances in Three-Dimensional Television and Video.

[Rogmans et al., 2010a] Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Biological-Aware Stereoscopic Rendering in Free Viewpoint Technology using GPU Computing. In *Proceedings of the 2010 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON '10)*, pages 1–4, Tampere, Finland, June 2010a.

[Rogmans et al., 2010b] Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Immersive GPU-Driven Biological Adaptive Stereoscopic Rendering. In *Proceedings of the 2010 3D Stereo Media Conference (3DSM '10)*, Liège, Belgium, December 2010b.

[Rost et al., 2009] Randi J. Rost, Bill M. Licea-Kane, Dan Ginsburg, John M. Kessenich, Barthold Lichtenbelt, Hugh Malan, and Mike Weiblen. *OpenGL Shading Language*. Addison-Wesley Professional, 3rd edition, July 2009. ISBN 978-032-16376-3-5.

[Ryoo et al., 2008] Shane Ryoo, Christopher Rodrigues, Sara Baghsorkhi, Sam Stone, David Kirk, and Wen-Mei Hwu. Optimization Principles and Application Performance Evaluation of a Multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '08)*, pages 73–82, Salt Lake City, UT, USA, February 2008.

[Sajadi and Majumder, 2010] Behzad Sajadi and Aditi Majumder. Auto-Calibration of Cylindrical Multi-Projector Systems. In *Proceedings of the 2010 IEEE Virtual Reality Conference (VR '10)*, pages 155–162, Waltham, MA, USA, March 2010.

[Sanders and Kandrot, 2010] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 1st edition, July 2010. ISBN 978-013-13876-8-3.

[Scharstein, 1996] Daniel Scharstein. Stereo Vision for View Synthesis. In *Proceedings of the 1996 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 852–858, San Francisco, CA, USA, June 1996.

[Scharstein and Szeliski, 2002] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision (IJCV)*, 47(1–3):7–42, April 2002.

[Scharstein and Szeliski, 2003] Daniel Scharstein and Richard Szeliski. High-Accuracy Stereo Depth Maps using Structured Light. In *Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, volume 1, pages 195–202, Madison, WI, USA, June 2003.

[Schmidt and Bannon, 1992] Kjeld Schmidt and Liam Bannon. Taking CSCW Seriously. *Computer Supported Cooperative Work (CSCW)*, 1(1–2):7–40, March 1992.

[Schreer and Sheppard, 2000] Oliver Schreer and Phil Sheppard. VIRTUE – The Step Towards Immersive Telepresence in Virtual Video-Conference Systems. In *Proceedings of the eBusiness and eWork Conference and Exhibition*, Madrid, Spain, October 2000.

[Schreer et al., 2001] Oliver Schreer, Nicole Brandenburg, Serap Askar, and Emanuele Trucco. A Virtual 3D Video-Conference System Providing Semi-Immersive Telepresence: A Real-Time Solution in Hardware and Software. In *Proceedings of the eBusiness and eWork Conference and Exhibition*, pages 184–190, Venice, Italy, October 2001.

[Schreer et al., 2008a] Oliver Schreer, Roman Englert, Peter Eisert, and Ralf Tanger. Real-Time Vision and Speech Driven Avatars for Multimedia Applications. *IEEE Transactions on Multimedia*, 10(3):352–360, April 2008a.

[Schreer et al., 2008b] Oliver Schreer, Ingo Feldmann, Nicole Atzpadin, Peter Eisert, Peter Kauff, and Harm Belt. 3DPresence – A System Concept for Multi-User and Multi-Party Immersive 3D Videoconferencing. In *Proceedings of the 5th European Conference on Visual Media Production (CVMP '08)*, pages 1–8, London, UK, November 2008b.

[Segal and Akeley, 1999] Mark Segal and Kurt Akeley. The OpenGL Graphics System: A Specification (Version 1.2.1). `http://www.opengl.org/documentation/specs/version1.2/OpenGL_spec_1.2.1.pdf`, April 1999.

[Seitz and Dyer, 1997] Steven M. Seitz and Charles R. Dyer. View Morphing: Uniquely Predicting Scene Appearance from Basis Images. In *Proceedings of the DARPA Image Understanding Workshop*, pages 881–887, New Orleans, LA, USA, May 1997.

[Shade et al., 1998] Jonathan Shade, Steven Gortler, Li-Wei He, and Richard Szeliski. Layered Depth Images. In *Proceedings of the 25th ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 231–242, Orlando, FL, USA, July 1998.

[Shechtman et al., 2005] Eli Shechtman, Yaron Caspi, and Michal Irani. Space-Time Super-Resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(4):531–545, April 2005.

[Shi and Tomasi, 1994] Jianbo Shi and Carlo Tomasi. Good Features to Track. In *Proceedings of the 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pages 593–600, Seattle, WA, USA, June 1994.

[Shreiner, 2009] Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. Addison-Wesley Professional, 7th edition, July 2009. ISBN 978-032-15526-2-4. The Khronos OpenGL ARB Working Group.

[Shum and Kang, 2000] Harry Shum and Sing B. Kang. Review of Image-Based Rendering Techniques. In *Proceedings of SPIE Visual Communications and Image Processing (VCIP '00)*, pages 2–13, Perth, WA, Australia, June 2000.

[Shum et al., 2007] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-Based Rendering*. Springer-Verlag New York, 1st edition, 2007. ISBN 978-038-72111-3-8.

[Sinha et al., 2004] Sudipta N. Sinha, Marc Pollefeys, and Leonard McMillan. Camera Network Calibration from Dynamic Silhouettes. In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*, volume 1, pages 195–202, Washington, DC, USA, June 2004.

[Slabaugh et al., 2002] Greg Slabaugh, Ron Schafer, and Mat Hans. Image-Based Photo Hulls. In *Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT '02)*, pages 704–709, Padova, Italy, June 2002.

[SMIT] SMIT. Studies Media Information Telecommunication. `http://smit.vub.ac.be`.

[Stankiewicz et al., 2013] Olgierd Stankiewicz, Krzysztof Wegner, Masayuki Tanimoto, and Marek Domanski. Enhanced Depth Estimation Reference Software (DERS) for Free-Viewpoint Television. Technical Report ISO/IEC JTC1/SC29/WG11 M31518, MPEG, Geneva, Switzerland, October 2013.

[Starck and Hilton, 2003] Jonathan Starck and Adrian Hilton. Model-Based Multiple View Reconstruction of People. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, pages 915–922, Nice, France, October 2003.

[Stone et al., 2010] John E. Stone, David Gohara, and Guochun Shi. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *IEEE Computing in Science & Engineering*, 12(3):66–73, May 2010.

[Strecha et al., 2004] Christoph Strecha, Rik Fransens, and Luc Van Gool. Wide-Baseline Stereo from Multiple Views: A Probabilistic Account. In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*, volume 1, pages I:552–I:559, Washington, DC, USA, June 2004.

[Sturm and Triggs, 1996] Peter F. Sturm and Bill Triggs. A Factorization Based Algorithm for Multi-Image Projective Structure and Motion. In *Proceedings of the 4th European Conference on Computer Vision (ECCV '96)*, volume II, pages 709–720, Cambridge, UK, April 1996.

[Su and He, 2011] Huihuang Su and Bingwei He. Stereo Rectification of Calibrated Image Pairs based on Geometric Transformation. *International Journal of Modern Education and Computer Science (IJMECS)*, 3(4):17–24, July 2011.

[Sun et al., 2003] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo Matching using Belief Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(7):787–800, July 2003.

[Sun et al., 2005] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric Stereo Matching for Occlusion Handling. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, volume 2, pages 399–406, San Diego, CA, USA, June 2005.

[Svoboda, 2000] Tomáš Svoboda. *Central Panoramic Cameras Design, Geometry, Egomotion*. PhD Thesis, Center for Machine Perception, Czech Technical University, Prague, Czech Republic, April 2000.

[Svoboda et al., 2002] Tomáš Svoboda, Hanspeter Hug, and Luc Van Gool. ViRoom - Low Cost Synchronized Multicamera System and Its Self-Calibration. In *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, pages 515–522, Zurich, Switzerland, September 2002.

[Svoboda et al., 2005] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A Convenient Multi-Camera Self-Calibration for Virtual Environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[Szeliski, 1996] Richard Szeliski. Video Mosaics for Virtual Environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, March 1996.

[Szeliski, 2010] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer-Verlag London, 1st edition, November 2010. ISBN 978-184-88293-4-3.

[Tanimoto et al., 2009] Masayuki Tanimoto, Toshiaki Fujii, and Kazuyoshi Suzuki. View Synthesis Algorithm in View Synthesis Reference Software 2.0 (VSRS 2.0). Technical Report ISO/IEC JTC1/SC29/WG11 M16090, MPEG, Lausanne, Switzerland, February 2009.

[Tao and Sawhney, 2000] Hai Tao and Harpreet S. Sawhney. Global Matching Criterion and Color Segmentation Based Stereo. In *Proceedings of the 5th IEEE Workshop on Applications of Computer Vision (WACV '00)*, pages 246–253, Palm Springs, CA, USA, December 2000.

[TinkerTouch, 2010] TinkerTouch. xplore it your way. `http://www.tinkertouch.com`, September 2010.

[Tobii Technology] Tobii Technology. Eye Tracker. `http://www.tobii.com`. World leader in eye tracking and gaze interaction.

[Tomasi and Manduchi, 1998] Carlo Tomasi and Roberto Manduchi. Bilateral Filtering for Gray and Color Images. In *Proceedings of the 6th IEEE International Conference on Computer Vision (ICCV '98)*, pages 839–846, Bombay, India, January 1998.

[Tombari et al., 2007] Federico Tombari, Stefano Mattoccia, and Luigi Di Stefano. Segmentation-Based Adaptive Support for Accurate Stereo Correspondence. In Domingo Mery and Luis Rueda, editors, *Advances in Image and Video Technology*, volume 4872 of *Lecture Notes in Computer Science (LNCS)*, pages 427–438. Springer Berlin Heidelberg, 2007.

[Tombari et al., 2008] Federico Tombari, Stefano Mattoccia, Luigi Di Stefano, and Elisa Addimanda. Near Real-Time Stereo Based on Effective Cost Aggregation. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR '08)*, pages 1–4, Tampa, FL, USA, December 2008.

[Tsai, 1987] Roger Y. Tsai. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology using Off-The-Shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[Tuytelaars and Van Gool, 2000] Tinne Tuytelaars and Luc Van Gool. Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions. In *Proceedings of the 11th British Machine Vision Conference (BMVC '00)*, pages 412–425, Bristol, UK, September 2000.

[Umeyama, 1991] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(4):376–380, April 1991.

[van Nimwegen, 2008] Christof van Nimwegen. Sociability Evaluation Report of the Demonstrators. IBBT ISBO VIN Project Deliverable 4.4.1, 2008.

[Vanaken, 2011] Cedric Vanaken. *Video-Based Animation: Articulated Video Sprites and Augmented Panoramic Video*. PhD Thesis, Hasselt University, Hasselt, Belgium, May 2011.

[Vanaken et al., 2008] Cedric Vanaken, Chris Hermans, Tom Mertens, Fabian Di Fiore, Philippe Bekaert, and Frank Van Reeth. Strike a Pose: Image-Based Pose Synthesis. In *Proceedings of the 13th International Workshop on Vision, Modeling, and Visualization (VMV '08)*, pages 131–138, Konstanz, Germany, October 2008.

[Vanloffeld et al., 2008] Rembert Vanloffeld, Maarten Dumont, Yannick Francken, and Philippe Bekaert. Real-Time Detectie en Tracking voor Sportanalyse met behulp van CUDA. Master's Thesis, Hasselt University, Hasselt, Belgium, 2008.

[Veksler, 2003] Olga Veksler. Fast Variable Window for Stereo Correspondence using Integral Images. In *Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, volume 1, pages 556–561, Madison, WI, USA, June 2003.

[Velizhev, 2005] Alexander Velizhev. GML C++ Camera Calibration Toolbox. `http://graphics.cs.msu.ru/en/node/909`, 2005.

[Vertegaal et al., 2003] Roel Vertegaal, Ivo Weevers, Changuk Sohn, and Chris Cheung. GAZE-2: Conveying Eye Contact in Group Video Conferencing using Eye-Controlled Camera Direction. In *Proceedings of the 21th SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*, pages 521–528, Fort Lauderdale, FL, USA, April 2003.

[Vetter, 1998] Thomas Vetter. Synthesis of Novel Views from a Single Face Image. *International Journal of Computer Vision (IJCV)*, 28(2):103–116, June 1998.

[Waizenegger et al., 2012] Wolfgang Waizenegger, Nicole Atzpadin, Oliver Schreer, Ingo Feldmann, and Peter Eisert. Model Based 3D Gaze Estimation for Provision of Virtual Eye Contact. In *Proceedings of the 19th IEEE International Conference on Image Processing (ICIP '12)*, pages 1973–1976, Orlando, FL, USA, September 2012.

[Wang et al., 2006a] Liang Wang, Mingwei Gong, Minglun Gong, and Ruigang Yang. How Far Can We Go with Local Optimization in Real-Time Stereo Matching. In *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '06)*, pages 129–136, Chapel Hill, NC, USA, June 2006a.

[Wang et al., 2006b] Liang Wang, Miao Liao, Minglun Gong, Ruigang Yang, and David Nister. High-Quality Real-Time Stereo Using Adaptive Cost Aggregation and Dynamic Programming. In *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '06)*, pages 798–805, Chapel Hill, NC, USA, June 2006b.

[Wang and Zheng, 2008] Zeng-Fu Wang and Zhi-Gang Zheng. A Region Based Stereo Matching Algorithm using Cooperative Optimization. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pages 1–8, Anchorage, AK, USA, June 2008.

[Waschbüsch et al., 2007] Michael Waschbüsch, Stephan Würmlin, and Markus Gross. 3D Video Billboard Clouds. *Computer Graphics Forum*, 26(3):561–569, September 2007.

[Wegner et al., 2013] Krzysztof Wegner, Olgierd Stankiewicz, Masayuki Tanimoto, and Marek Domanski. Enhanced View Synthesis Reference Software (VSRS) for Free-viewpoint Television. Technical Report ISO/IEC JTC1/SC29/WG11 M31520, MPEG, Geneva, Switzerland, October 2013.

[Weissig et al., 2005] Christian Weissig, Ingo Feldmann, Joachim Schüssler, Ulrich Höfker, Peter Eisert, and Peter Kauff. A Modular High-Resolution Multi-Projection System. In *Proceedings of the 2nd Workshop on Immersive Communication and Broadcast Systems (ICOB '05)*, Berlin, Germany, October 2005.

[Weissig et al., 2012] Christian Weissig, Oliver Schreer, Peter Eisert, and Peter Kauff. The Ultimate Immersive Experience: Panoramic 3D Video Acquisition. In Klaus Schoeffmann, Bernard Merialdo, Alexander G. Hauptmann, Chong-Wah Ngo, Yiannis Andreopoulos, and Christian Breiteneder, editors, *Advances in Multimedia Modeling*, volume 7131 of *Lecture Notes in Computer Science (LNCS)*, pages 671–681. Springer Berlin Heidelberg, 2012. ISBN 978-364-22735-4-4.

[Wobbrock et al., 2009] Jacob O. Wobbrock, Meredith R. Morris, and Andrew D. Wilson. User-Defined Gestures for Surface Computing. In *Proceedings of the 27th SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*, pages 1083–1092, Boston, MA, USA, April 2009.

[Wolff et al., 2007] Robin Wolff, Dave J. Roberts, Anthony Steed, and Oliver Otto. A Review of Telecollaboration Technologies with Respect to Closely Coupled Collaboration. *International Journal of Computer Applications in Technology (IJCAT)*, 29(1): 11–26, January 2007.

[Wu et al., 2008] Wanmin Wu, Zhenyu Yang, Indranil Gupta, and Klara Nahrstedt. Towards Multi-Site Collaboration in 3D Tele-Immersive Environments. In *Proceedings of the*

*28th International Conference on Distributed Computing Systems (ICDCS '08)*, pages 647–654, Beijing, China, June 2008.

[Xiong and Turkowski, 1997] Yalin Xiong and K. Turkowski. Creating Image-Based VR using a Self-Calibrating Fisheye Lens. In *Proceedings of the 1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 237–243, San Juan, Puerto Rico, June 1997.

[Yang, 2014] Qingxiong Yang. Hardware-Efficient Bilateral Filtering for Stereo Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36 (5):1026–1032, May 2014.

[Yang et al., 2006a] Qingxiong Yang, Liang Wang, Ruigang Yang, Shengnan Wang, Miao Liao, and David Nister. Real-Time Global Stereo Matching using Hierarchical Belief Propagation. In *Proceedings of the 17th British Machine Vision Conference (BMVC '06)*, volume 6, pages 989–998, Edinburgh, UK, September 2006a.

[Yang et al., 2008] Qingxiong Yang, Chris Engels, and Amir Akbarzadeh. Near Real-time Stereo for Weakly-Textured Scenes. In *Proceedings of the 19th British Machine Vision Conference (BMVC '08)*, pages 72.1–72.10, Leeds, UK, September 2008.

[Yang et al., 2009] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewenius, and David Nistér. Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(3):492–504, March 2009.

[Yang et al., 2010a] Qingxiong Yang, Liang Wang, and Narendra Ahuja. A Constant-Space Belief Propagation Algorithm for Stereo Matching. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pages 1458–1465, San Francisco, CA, USA, June 2010a.

[Yang and Pollefeys, 2003] Ruigang Yang and Marc Pollefeys. Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In *Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 211–220, Madison, WI, USA, June 2003.

[Yang and Welch, 2002] Ruigang Yang and Greg Welch. Fast Image Segmentation and Smoothing using Commodity Graphics Hardware. *Journal of Graphics Tools*, 7(4): 91–100, December 2002. Special Issue on Hardware-Accelerated Rendering Techniques.

[Yang and Zhang, 2002] Ruigang Yang and Zhengyou Zhang. Eye Gaze Correction with Stereovision for Video-Teleconferencing. In *Proceedings of the 7th European Conference on Computer Vision (ECCV '02)*, volume II, pages 479–494, Copenhagen, Denmark, May 2002.

[Yang et al., 2002] Ruigang Yang, Greg Welch, and Gary Bishop. Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications (PG '02)*, pages 225–234, Beijing, China, October 2002.

[Yang et al., 2004a] Ruigang Yang, Marc Pollefeys, and Sifang Li. Improved Real-Time Stereo on Commodity Graphics Hardware. In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '04)*, pages 36–36, Washington, DC, USA, June 2004a.

[Yang et al., 2004b] Ruigang Yang, Marc Pollefeys, Hua Yang, and Greg Welch. A Unified Approach to Real-Time, Multi-Resolution, Multi-Baseline 2D View Synthesis and 3D Depth Estimation using Commodity Graphics Hardware. *International Journal of Image and Graphics (IJIG)*, 4(4):627–651, October 2004b.

[Yang et al., 2005] Zhenyu Yang, Klara Nahrstedt, Yi Cui, Bin Yu, Jin Liang, Sang-Hack Jung, and Ruzena Bajscy. TEEVE: The Next Generation Architecture for Tele-Immersive Environments. In *Proceedings of the 7th IEEE International Symposium on Multimedia (ISM '05)*, pages 112–119, Irvine, CA, USA, December 2005.

[Yang et al., 2006b] Zhenyu Yang, Bin Yu, Klara Nahrstedt, and Ruzena Bajscy. A Multi-Stream Adaptation Framework for Bandwidth Management in 3D Tele-Immersion. In *Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSDAV '06)*, pages 14:1–14:6, Newport, RI, USA, May 2006b.

[Yang et al., 2006c] Zhenyu Yang, Bin Yu, Wanmin Wu, Ross Diankov, and Ruzena Bajscy. Collaborative Dancing in Tele-Immersive Environment. In *Proceedings of the 14th ACM International Conference on Multimedia (MM '06)*, pages 723–726, Santa Barbara, CA, USA, October 2006c.

[Yang et al., 2010b] Zhenyu Yang, Wanmin Wu, Klara Nahrstedt, Gregorij Kurillo, and Ruzena Bajcsy. Enabling Multi-Party 3D Tele-Immersive Environments with ViewCast. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(2):7:1–7:30, March 2010b.

[Yoon and Kweon, 2006] Kuk-Jin Yoon and In-So Kweon. Adaptive Support-Weight Approach for Correspondence Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(4):650–656, April 2006.

[Zabulis et al., 2006] Xenophon Zabulis, Georgios Kordelas, Karsten Mueller, and Aljoscha Smolic. Increasing the Accuracy of the Space-Sweeping Approach to Stereo Reconstruction, using Spherical Backprojection Surfaces. In *Proceedings of the 13th IEEE*

*International Conference on Image Processing (ICIP '06)*, pages 2965–2968, Atlanta, GA, USA, October 2006.

[Zach et al., 2008] Christopher Zach, David Gallup, Jan-Michael Frahm, and Marc Niethammer. Fast Global Labeling for Real-Time Stereo using Multiple Plane Sweeps. In *Proceedings of the 13th International Workshop on Vision, Modeling, and Visualization (VMV '08)*, pages 243–252, Konstanz, Germany, October 2008.

[Zhang et al., 2009a] Ke Zhang, Jiangbo Lu, and Gauthier Lafruit. Cross-Based Local Stereo Matching using Orthogonal Integral Images. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 19(7):1073–1079, July 2009a.

[Zhang et al., 2009b] Ke Zhang, Jiangbo Lu, Gauthier Lafruit, Rudy Lauwereins, and Luc Van Gool. Real-Time Accurate Stereo with Bitwise Fast Voting on CUDA. In *Proceedings of the 12th IEEE International Conference on Computer Vision Workshops (ICCVW '09)*, pages 794–800, Kyoto, Japan, September 2009b.

[Zhang et al., 2011a] Ke Zhang, Jiangbo Lu, Qiong Yang, Gauthier Lafruit, Rudy Lauwereins, and Luc Van Gool. Real-Time and Accurate Stereo: A Scalable Approach with Bitwise Fast Voting on CUDA. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 21(7):867–878, July 2011a.

[Zhang et al., 2011b] Lu Zhang, Ke Zhang, Tian Sheuan Chang, Gauthier Lafruit, Georgi Krasimirov Kuzmanov, and Diederik Verkest. Real-Time High-Definition Stereo Matching on FPGA. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '11)*, pages 55–64, Monterey, CA, USA, February 2011b.

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

[Zhang, 2012] Zhengyou Zhang. Microsoft Kinect Sensor and its Effect. *IEEE MultiMedia*, 19(2):4–10, February 2012.

[Zitnick and Kang, 2007] C. Lawrence Zitnick and Sing Bing Kang. Stereo for Image-Based Rendering using Image Over-Segmentation. *International Journal of Computer Vision (IJCV)*, 75(1):49–65, October 2007.

[Zitnick et al., 2004] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-Quality Video View Interpolation using a Layered Representation. *ACM Transactions on Graphics (TOG)*, 23(3):600–608, August 2004.