Improved Real-Time Rendering for Mixed Reality

David R. Walton

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Engineering

of

University College London.

Department of Computer Science

University College London

May 16, 2019

I, David R. Walton, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The research presented in this thesis explores methods for rendering realistic virtual content in mixed reality applications. Rendering content for mixed reality presents unique challenges, as it is critical that the virtual content is not only realistic, but also consistent with its real surroundings. This thesis is concerned with methods to achieve this goal, focusing on those which require minimal prior information about the real scene and work in real time. In particular, methods are presented which address two problems in this area.

The first problem is correct handling of occlusion between virtual and real objects. When adding virtual content to a real scene, it is challenging to determine where real objects should occlude the added virtual content. An approach was developed to combine the noisy colour and depth outputs of RGBD cameras to determine accurate real-virtual occlusions. A method was also developed to quantitatively assess the quality of such approaches.

The second problem is capturing a detailed lighting model of a real environment quickly, and updating it in real time. The appearance of objects is greatly dependent on the surrounding lighting environment, so a detailed lighting model is invaluable when attempting to render realistic virtual content into the scene. A number of novel approaches to capture this lighting information and use it are developed, and applications for these approaches are also explored, including rendering virtual objects which reflect the changing real world around them. In contrast to previous approaches which require external light probes or infer lighting indirectly from the real scene, the presented approaches use a self-contained two-camera system and use the extra information to infer the lighting at the virtual object location.

Impact Statement

The research presented in this thesis explores new methods for capturing and using data from the real world to improve rendering for mixed reality (MR). In an academic setting, this contributes to an extensive body of work carried out by researchers over a number of years on mixed reality and realistic rendering techniques. The research has been presented at international, peer reviewed VR and MR conferences, exposing it to the wider academic community.

During the thesis, focus was placed on exploring methods which used consumer hardware, and did not require expert knowledge or a long time investment in carrying out precapture steps. The intent was to develop techniques which could form part of MR systems accessible to as many people as possible. It is the author's hope that as the necessary hardware becomes cheaper and more widely available, techniques such as these will form part of compelling MR applications used by a wide range of people.

The occlusion handling method presented here addresses an important problem in many MR applications; that real content does not usually occlude virtual content. This makes the virtual content appear unrealistic and confuses users about the spatial relationships between real and virtual objects. The technique tackles these problems, and critically handles the case where a user's hands occlude a virtual object. This opens up the possiblity in the future of users being able to pick up and manipulate virtual objects in MR via gestures, or even using haptic feedback gloves.

The lighting capture methods provide ways to extract more information about the real scene, which is useful for adding virtual objects which interact with the real

lighting and geometry. As is shown in the thesis, this can be used to allow virtual objects to cast shadows on real objects, reflect nearby real objects, and feel like a cohesive part of a real scene. In the future, these techniques could be developed further and combined with new rendering approaches to offer truly photorealistic mixed reality experiences.

Acknowledgements

This research was supported by the EPSRC and Imagination Technologies Ltd.

I would first like to thank my primary supervisor, Anthony Steed. I am extemely grateful for his help throughout the project, and offering his depth of experience in the area. I am also very grateful to my second supervisor, Simon Julier, for our valuable discussions and his feedback on the research. My thanks also to Tobias Ritschel for his advice and input.

I would also like to thank Imagination Technologies, and particularly my industrial supervisors Paul Brasnett and Luke Peterson. I really appreciated their enthusiasm for the project, and all their helpful advice. I'd also like to thank all the wonderful people I've worked with in PowerVR research (too many to mention here).

My thanks also to all the people in the VR group at UCL who made it such a great place to be: Sebastian Friston, Ye Pan, David Swapp, Andrew MacQuarrie, María Murcia Lopez, Jacob Thorn and Ben Congdon, as well as our friends from the VEIV group, James Hennessy, Lucy Zarina Campbell and Kelvin Wong. I'm pretty sure doing a doctorate isn't supposed to be fun, but you all made it fantastic.

I'm very grateful for the supervision of Akihiro Sugimoto during my internship at the NII, especially for our valuable meetings and discussions with Diego Thomas. I'd also like to thank all of Prof. Sugimoto's students, particularly Fumiki Sekiya for making the NII so welcoming.

Finally, my thanks to my mother, father and sister for all their love and support throughout the thesis - I couldn't have done it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis focuses on improved rendering techniques for mixed reality (MR). Mixed reality refers to the blending of stimuli from the real world with those produced by a computer to produce a user experience combining virtual and real content. Mixed reality approaches may add virtual stimuli, or mediate and modify stimuli from the real world.

Milgram et al. [99] define a Reality-Virtuality Continuum. This continuum ranges from purely real environments to purely virtual environments. Environments lying within this continuum contain both real and virtual content, and are referred to as Mixed Reality (MR) environments. The more specific terms Augmented Reality (AR) and Augmented Virtuality (AV) are used to refer to environments containing mainly real objects or mainly virtual objects respectively.

Mixed reality environments can be experienced on a range of different platforms. These can be broadly classified into video see-through and optical see-through displays. Optical see-through displays allow some or all of the light from the real scene to pass through to the user's eyes, augmenting it with light to give the impression that additional virtual content is present in the scene. Video see through displays instead capture the real scene using a camera or cameras, modify or process the camera output to add virtual content, and then display this information to the user.

In this thesis, the focus will be on video see-through systems, and all examples shown will use such systems. Many of the presented techniques would also be

applicable to optical see-through systems as well, however.

The thesis will mainly focus on MR applications in which the virtual content is intended to appear to be a part of the real environment. In addition to the challenges involved in rendering realistic virtual content in real time, this presents additional challenges in making the content appear consistent with the real content around it.

This thesis attempts to tackle some of these challenges. In particular, the focus is on the problems of correctly handling real-virtual occlusion, and capturing real-world illumination in real time. This captured illumination can be used to correctly render photorealistic virtual content and simulate its effects on the surrounding real environment.

1.1 Research Problem

Differential rendering [28] provides in principle a solution to the problem of rendering virtual objects into real scenes in a believable way, making the objects appear to be illuminated by the environment around them, and also rendering the influence of the virtual objects on their surroundings. In practice, however, differential rendering techniques require information about the real world which is generally not available - specifically, geometry and material property information. This thesis will focus on ways to improve the rendering of virtual objects in MR, which are tolerant to information about the real environment being incomplete or inaccurate.

More specifically, the thesis focuses on two specific areas. The first is the correct handling of real-virtual occlusion, and is discussed in part I of the thesis. The second, capturing illumination models of real environments without light probes, is discussed in part II.

1.2 Research Questions & Scope

Part I of the thesis explores how RGBD cameras can be used to improve occlusion handling in mixed reality applications. These cameras capture valuable real depth information, but this information is noisy and incomplete. The question is how this depth data can be processed and combined with the RGB data to provide accurate MR occlusion.

Within this general question, a number of areas were explored. First, approaches were developed which took the colour and depth frame pairs supplied by the RGBD camera as input, and processed these to estimate accurate occlusion. Following this, the quality of the approaches was quantitatively assessed, using a novel comparison approach.

Part II of the thesis focuses on a separate but related problem - how can a detailed, accurate lighting model of the real world be captured and updated in real time, and how can this be effectively used to render realistic virtual content for MR?

Specifically, the use of self-contained systems of cameras to capture this information was investigated. This differed from much of the existing literature, which has tended to use external light probes or obtain less detailed information indirectly through inverse rendering approaches.

1.3 Structure

Chapter 2 gives an overview of literature relevant to this project. It gives a brief overview of mixed reality in general, and then covers more specific topics in this area: realistic rendering, tracking and reconstruction, occlusion handling and capturing real illumination information.

The remainder of the thesis is divided into two parts, the first relating to real-virtual occlusion, and the second to capturing real world illumination.

Chapter 3 in part I focuses on the problem of real-virtual occlusion - this involves determining at which pixels a virtual object should be occluded by real objects in the scene, and hence should not be rendered.

The subsequent chapters in part II focus on capturing real world environment illumination without the use of light probes, and explore some of the ways in which this information can be used to render realistic virtual content into MR scenes.

Chapter 4 details the first approach developed in the course of the EngD project to tackle this problem. This early approach was able to render specular virtual objects with detailed reflections of their real surroundings, but was limited to capturing only low dynamic range information.

Chapter 5 builds on chapter 4, adapting the approach to capture and use high dynamic range information from the real scene.

Chapter 6 focuses on the development of a final system which improves upon the previous approaches in a number of ways, most notably in requiring no prior knowledge about the real environment geometry.

1.4 Contributions

1.4.1 Methodological Contributions

1. A method for evaluating the quality of per-frame quality and temporal stability of AR occlusion methods relative to ground truth (Chapter 3).

1.4.2 Technical Contributions

1. Design of an approach for rendering accurate real-virtual occlusion in video see-through MR applications, given noisy and/or incomplete colour & depth inputs (Chapter 3).

2. Design of an approach for capturing and updating in real time a detailed illumination model for the real environment in an MR application (Chapter 4).

3. Extension of the previous approach to capture photometrically accurate, high dynamic range lighting information (Chapter 5).

4. Development of an approach for capturing a similar illumination model using no prior geometric information about the real environment (Chapter 6).

1.4.3 Publications

This project has resulted in the following publications, appearing in peer-reviewed conferences:

WALTON, D.R., THOMAS, D., STEED, A., SUGIMOTO, A. Synthesis of Environment Maps for Mixed Reality. In *16th International Symposium on Mixed and Augmented Reality, ISMAR 2017* (2017)
Contains extracts of work presented in Chapter 4.

WALTON, D.R., STEED, A. Accurate Real-time Occlusion for Mixed Reality. In *23rd ACM Symposium on Virtual Reality Software and Technology, VRST 2017* (2017)

Contains extracts of work presented in Chapter 3.

WALTON, D.R., STEED, A. Dynamic Environment Capture for Mixed Reality. In *24th ACM Symposium on Virtual Reality Software and Technology, VRST 2018* (2018)

Contains extracts of work presented in Chapter 6.

Chapter 2

Background

2.1 Mixed Reality

Virtual reality systems attempt to give the user the experience of entering a purely virtual, computer-generated environment. In virtual reality applications, the aim is to replace the real sensory inputs of a user with synthetic ones, using devices such as VR headsets, headphones and haptic feedback devices. Other systems have a different focus, choosing to selectively replace or modify the user's sensory inputs, allowing them to experience a mixed environment consisting of virtual and real content. This concept is known as mixed reality (MR).

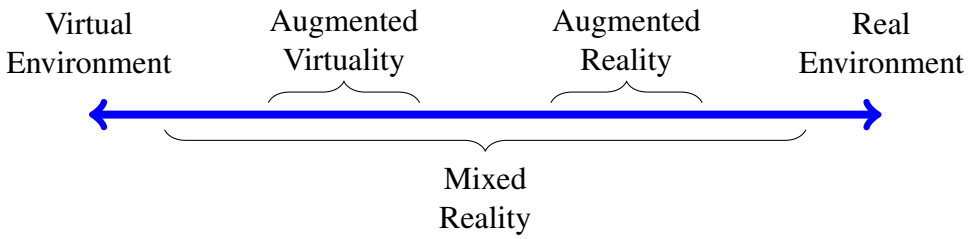


Figure 2.1: Representation of a reality-virtuality continuum. Based upon a diagram from [98, 99].

More precisely, Milgram et al. [98, 99] introduced the concept of mixed reality spanning a reality-virtuality continuum, ranging from reality to virtual reality (see figure 2.1). As one moves along this line, a larger proportion of a user's real sensory inputs are replaced with synthesised, virtual stimuli. Other terms used in this area such as "augmented reality" (AR) and "augmented virtuality" (AV) can be viewed as encompassing parts of this continuum. The term X-reality (cross-reality, or ex-

tended reality), or "XR" has a similar meaning to MR, but has a more expansive definition, with more focus on interactions between the virtual and real domains.

In addition to the usual challenges present when rendering realistic virtual content in purely virtual scenes, rendering realistic virtual content for MR applications presents some unique difficulties. Hardware must be developed capable of displaying virtual and real content to the user simultaneously. The rendered content must not only appear realistic in isolation, but must also be consistent with the real content around it. In particular, the content must be correctly registered to the real environment, must be occluded by real content where appropriate, and must appear to be illuminated by the real environment around it. Furthermore, the influence of the virtual content on its real surroundings must also be accounted for - for example, an opaque virtual object placed on a real table might be expected to cast a shadow.

This challenge relates to the Extent of World Knowledge (EWK) axis in Milgram et al.'s taxonomy of MR displays [98], a representation of which is shown in figure 2.2. This classifies MR applications according to the extent to which they have access to knowledge of the real environment. At the extreme left end of this scale, the application has no knowledge of the real environment at all; not only is the application unaware of the geometry and properties of the real scene, it also lacks information about the location of the display in the real world. At the extreme right end of the scale, the application has access to a full, detailed model of the real environment, sufficient to render it in a VR setting, in addition to knowing the location of the display relative to this model.

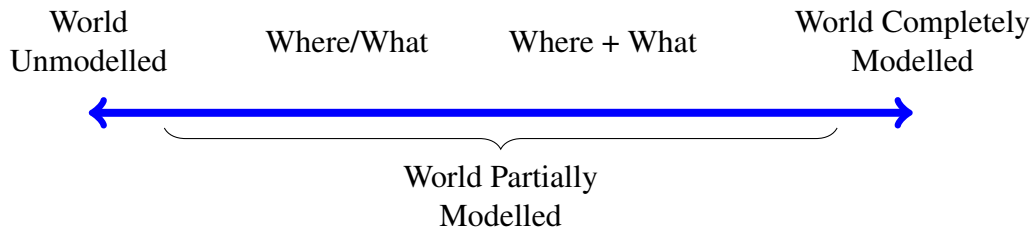


Figure 2.2: Extent of World Knowledge (EWK) in mixed reality applications. Based upon a diagram from [98].

Most MR applications sit somewhere between these two extremes, having access to only partial information about the real environment. For example, an appli-

cation based on tracking of fiducial markers, such as that developed by Kato and Billinghurst [73] only has knowledge of the pose of the display relative to each fiducial marker. In particular, it only has access to the "Where" information shown to the left of figure 2.2. Recently, approaches such as Kinect Fusion [106] have enabled detailed models of the real environment to be reconstructed in real time, whilst simultaneously tracking the camera. This provides the "Where + What" information shown towards the right of the axis. The greater EWK enables a wider range of interesting interactions between virtual and real objects, such as the physics simulation between virtual balls and the real environment shown in [106].

The work presented in this thesis attempts to provide ways to develop MR applications with a greater EWK, enabling these new interactions between virtual and real content. It particularly focuses on developing EWK when starting with little prior knowledge of the real environment.

In this section, existing literature relating to MR applications is discussed. Section 2.2 first explores MR display hardware. Section 2.3 then explores how the content driving these displays is rendered. Sections 2.4, 2.5 and 2.6 discuss issues relating to the development of EWK. Section 2.4 explores work related to tracking motion and building a model of the real world in an MR application. Section 2.5 discusses the more specific problem of determining where virtual content should be occluded by real content. Finally, section 2.6 explores works which attempt to capture knowledge about the light sources illuminating the real environment.

2.2 Mixed Reality Display Devices

Mixed reality applications require a method for displaying a mixture of both real and virtual content to a user simultaneously. There are a wide variety of display devices which can be used to achieve this.

Broadly, MR display devices can be categorised as either optical see-through or video see-through.

Optical see-through devices allow light from the real world to pass through the display, so that the real world is directly visible to the user. Such devices modify

the light passing through them in order to add virtual content. The first optical see-through HMD was developed by Sutherland [133], and was nicknamed the "Sword of Damocles" due to the large ceiling-mounted mechanical tracking system. At the time of writing, optical see-through devices including the Microsoft Hololens[1] and Magic Leap One[2] are becoming commercially available, which are compact and self-contained, capable of tracking using on-board sensors.

Video see-through devices capture the real world via one or more imaging devices, modify the resulting images to add virtual content, and then finally display the captured images to the user on a conventional opaque display. It is possible to implement video see-through MR systems on devices such as mobile phones or tablets, or alternatively via a HMD equipped with cameras such as the VRVANA[3] or AR Rift [132, 130]. Current AR SDKs such as ARKit[4] and ARCore[5] make it comparatively straightforward for developers to produce markerless AR apps working on the majority of modern mobile devices.

Other systems use display devices such as projectors to add virtual content at a specific location in a virtual environment. Such systems are sometimes referred to as Spatial Augmented Reality (SAR) systems [8].

2.2.1 Optical See-through Devices

Since optical see-through systems display the real world directly, they avoid issues related to degrading the quality of the user's perception of the real world. The real world observed by the user suffers from no latency, loss of resolution or loss of focal cues. This is an important advantage for a number of reasons - it improves the user experience, and reduces the possibility of issues such as simulator sickness. Optical see-through systems also have inherent safety advantages - the users of such systems are still capable of seeing the real world if the display should fail.

There are a number of challenges involved in making successful optical see-through MR systems however, particularly if the focus is on rendering realistic vir-

[1] https://www.microsoft.com/en-us/hololens
[2] https://www.magicleap.com/magic-leap-one
[3] http://vrvana.com/index.html
[4] https://developer.apple.com/arkit/
[5] https://developers.google.com/ar/

tual content. Optical see-through devices require displays which can be expensive and complicated to produce, increasing the cost of such devices.

Some degree of latency will be involved in determining the pose of the device, rendering virtual content accordingly and then displaying this content to the user. If this latency is too great, the registration between real and virtual content will appear incorrect, particularly during rapid motion of the device. This can reduce the perceived realism of the virtual content, and in extreme cases make the device difficult to use [35].

When observing real objects, the shape of the lenses in the user's eyes is changed to bring them into focus. However, current commercial optical see-through displays can only display virtual content at a limited number of discrete focal depths (one in the Hololens, and two in the Magic Leap One). This causes the virtual object to potentially appear inconsistent with the real world, and could also make it difficult for the user to focus on virtual and real content simultaneously, even when they appear to be at the same depth. The problem is more pronounced when virtual objects are closer to the display.

Only a small number of optical see-through systems such as those of Kiyokawa et al. [77] and Gao et al. [41] are capable of selectively completely blocking incoming light from the real environment on parts of the display. For many systems, including the Hololens and Magic Leap One, it is not possible to display truly opaque virtual content - some light from the real environment behind the virtual object will always pass through, causing the virtual content to appear translucent. This greatly limits the ability of such devices to render virtual content which appears to be real, particularly in bright environments.

2.2.2 Video See-through Devices

In general, video see-through systems have the advantage of allowing virtual content to appear more similar to real content, but they degrade the user's ability to perceive the real world to some degree.

In contrast to optical see-through displays, both the rendered virtual content and displayed real content may suffer from latency. In the case of the virtual content,

the latency results from the need to determine the pose of the display via tracking, render the virtual content appropriately and then display it. For the real content, the latency results from needing to capture an image or images of the real world, process them appropriately and then display them. Consequently, the latency is often different for the virtual and real content. Excessive latency may reduce the user's ability to perceive both the virtual and real environments, in addition to causing issues such as "simulator sickness" in HMDs.

Video see-though MR systems are capable of rendering fully opaque virtual objects, and allow for differential rendering techniques to be used to produce realistic virtual objects to the environment. Furthermore, changes to the real environment such as adjusting real lighting and modifying real material properties are possible. Diminished reality techniques can also be applied to remove real objects from the scene. This opens up a wider range of possible applications for such devices.

Video see-through devices can use conventional displays such as those in VR headsets or on the screens of mobile devices, and do not require the more expensive specialised display hardware of optical see-through devices. However, suitable high-framerate, wide field of view cameras are required to capture the real world and specialised processing hardware may be needed to reduce the latency involved in capturing, processing and displaying the images.

2.3 Photorealistic Rendering for Mixed Reality

As briefly mentioned in section 2.1, rendering photorealistic content for mixed reality presents a number of challenges. This section will focus specifically on the problem of rendering realistic virtual content and its influence on the real environment, under the assumption that lighting information and real scene geometry and material information is available.

2.3.1 Photorealistic Rendering of Virtual Content

Many techniques used in realistic mixed reality rendering build on techniques intended for rendering purely virtual environments. Real-time realistic rendering is a vast research topic which can only be briefly introduced here.

2.3.1.1 The Rendering Equation

The appearance of a real object is a consequence of both its intrinsic material properties and the lighting environment around it. The light incident on a surface point is a combination of light emitted directly by light sources, in addition to light reflected, refracted, scattered etc. by other scene objects. This concept was formalised by Kajiya in the rendering equation [62]. For a point x on the surface of an opaque object, this can be expressed as follows:

$$L_o(x,\omega_o) = L_e(x,\omega) + \int_\Omega f(x,\omega_i,\omega_o) L_i(x,\omega_i)(\omega_i \cdot n_x) d\omega_i \tag{2.1}$$

Here, L_o is the radiance leaving x in the direction ω_o. L_e is the radiance emitted by the surface itself at x. The surface normal at x is n_x, and Ω is the hemisphere centred around n. L_i is the incident radiance. The function f is the bidirectional reflectance distribution function (BRDF), and captures the properties of the surface material wrt its response to incoming light.

There are a number of challenges involved in solving this equation in practice. One of these challenges is finding L_i, as this may consist in part of light reflected from other surface points, requiring the rendering equation to be solved there as well. This results in a recursive definition. This is clearer in an alternative expression of the rendering equation, which uses an integral over the scene surfaces S:

$$I(x,x') = g(x,x') \left(e(x,x') + \int_S \rho(x,x',x'') I(x',x'') dx'' \right) \tag{2.2}$$

In this equation, $I(x,x')$ is the intensity of light emitted from x' to x. The geometry term g is 1 if x' is visible from x, and 0 otherwise. e is the intensity of light emitted by the surface at x' to x. Finally, $\rho(x,x',x'')$ captures what proportion of the light emitted from x'' to x' is reflected on to x, and depends upon the BRDF at x' and position and orientation of x, x' and x''. These terms are defined more precisely in [62].

Here, I is defined recursively, with deeper recursion depth corresponding to light paths involving more bounces. Fortunately, since energy is lost each time

light is reflected, considering only a finite number of bounces is often sufficient to approximate the true result closely.

Realistic rendering techniques generally consist of attempting to solve the rendering equation in one of its forms, often with some form of simplification.

2.3.1.2 Global vs Local Illumination

Rendering methods can be broadly categorised into local illumination (LI) and global illumination (GI) approaches. LI approaches calculate the appearance of each surface point independently of the rest of the scene. Referring back to equation 2.1, this corresponds to assuming that L_i is fixed and does not depend upon L_o at other surface points. This removes the recursive element from the rendering equation, making solving it much simpler. In practice, further simplifications are often made such as modelling the lighting environment L_i as a number of simple point or directional sources, and using a simple reflectance model such as that proposed by Phong [112] to approximate the BRDF f and remove the need to perform a spherical integral at each surface point.

LI approaches are efficient and highly parallelisable, making them well-suited to real-time rendering approaches using modern massively parallel graphics hardware. They are, however, incapable of simulating many real lighting interactions where lighting at one point is affected other points in the real scene. The effects of such interactions include cast shadows, caustics and interreflection (colour bleeding). GI techniques attempt to simulate some of these interactions, and therefore provide more realistic results.

Some GI techniques can be viewed as attempting to solve the rendering equation numerically, for example via Monte Carlo methods (path-tracing) or finite element methods (radiosity). However, many real-time techniques focus on approximating a single GI phenomenon. Examples include shadow mapping, which is used to render shadows cast by opaque objects, and environment mapping, which can be used to render reflections from highly specular surfaces. A much more detailed review of real-time GI methods can be found in [115].

It is also possible to precompute GI using an offline approach, and store the

result for use at runtime. If the virtual scene and lighting environment are both fixed, the result can simply be stored in textures for the virtual objects, which are typically referred to as lightmaps. More flexible precomputation approaches allow the lighting environment or real scene geometry to change at runtime.

2.3.1.3 Precomputed Radiance Transfer

Precomputed radiance transfer (PRT) [127] is a technique for real-time photorealistic rendering which is used as part of the MR approaches detailed in chapters 4, 5 and 6 of this thesis. PRT is an approach which allows realistic GI lighting interactions to be precomputed and stored, enabling them to be rendered in real-time at runtime. The following is a brief overview of diffuse PRT - more detail, and a discussion of applying PRT to the specular case can be found in [127].

The overall approach of PRT is to store a mapping between the incident radiance and emitted radiance for each point on the surface of a virtual object. In this thesis, we focus on the diffuse case, where the emitted radiance is the same in all directions. Given this assumption, PRT involves precomputing a transfer function M so that the emitted radiance can be expressed as the integral $\int_{S_2} M(s)L(s)ds$. Here, L is the lighting environment, expressed as a spherical function, and S_2 is the sphere $\{x \in \mathbb{R}^3 \mid |x| = 1\}$. M encapsulates a number of components of the rendering equation 2.1; the BRDF f, the effect on L_i of reflections and shadowing by other surface points, and the dot product $\omega_i \cdot n_x$.

This presents a number of problems; M is a full function $S_2 \to \mathbb{R}$, so storing such a function for each surface point on an object would consume a large amount of memory. Furthermore, the integral needs to be computed at each visible surface point at each frame during runtime, which would be prohibitively expensive if it were computed using numerical integration.

PRT addresses both of these problems by projecting the functions M and L to a basis of functions $S_2 \to \mathbb{R}$. The basis used in [127] is the spherical harmonic (SH) basis, which can be viewed as analogue of the Fourier basis for spherical functions (the first few bands of SH functions are shown in figure 2.3). A spherical function can be projected to the first N bands of spherical harmonics, resulting in a projection

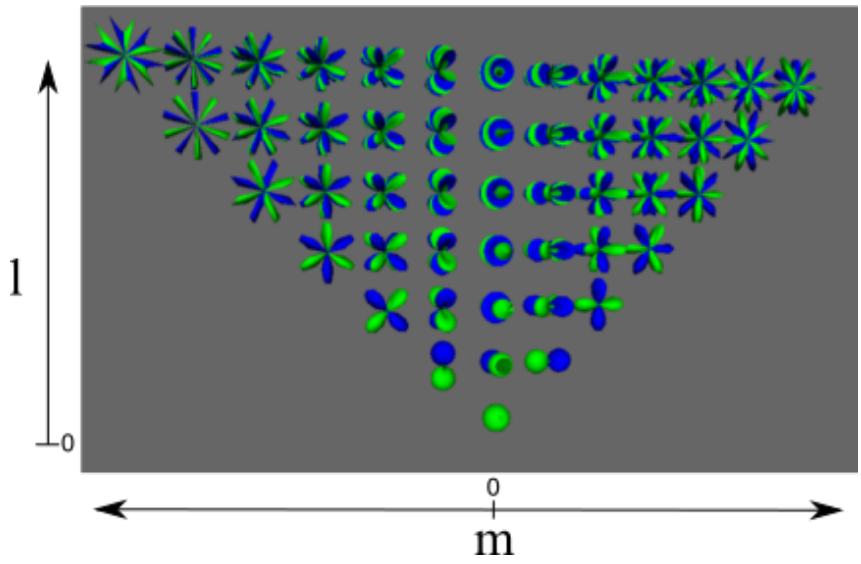


Figure 2.3: First 7 bands of SH basis functions, as spherical plots. In these plots, the distance from the centre represents magnitude, and the colour represents the sign (green is positive, and blue is negative). Image rendered by the author.

which can be stored using just N^2 coefficients, at the cost of some accuracy and the loss of higher frequencies. This allows the transfer functions for each surface point to be stored compactly. Furthermore, these projections have the following useful property:

$$\int_{S_2} \widetilde{A}(s)\widetilde{B}(s) = \sum_i A_i B_i$$

Here, A and B are spherical functions, $\widetilde{A}$ is the SH approximation to A and A_i is the i^{th} coefficient of the SH projection of A. This means that, at runtime, the lighting is simply calculated by taking the dot product of the incident lighting coefficients with the transfer coefficients at each visible surface point.

Note that this discussion has focused on grayscale, but PRT can be applied to colour as well by applying this technique independently to each colour channel. Figure 2.4 shows an example of a lighting environment (the spherical plot on the left of the image) and a virtual object illuminated by it via PRT. Here, grayscale PRT was used to determine shadow intensity, and the colour was determined by an albedo texture.

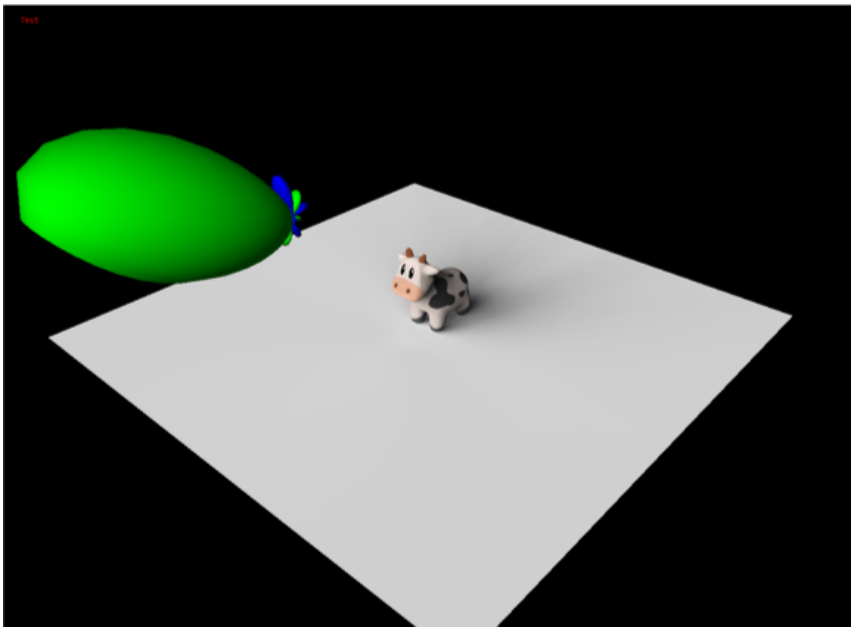


Figure 2.4: Example 3D cow object on planar surface, with lighting simulated via PRT. The SH projection of the incident radiance is shown as a blue/green spherical plot at the top left (plotted in the same way as the basis in figure 2.3). The 3D model used was created by Keenan Crane [23], and has been released to the public domain.

2.3.2 Differential Rendering

Debevec [28] introduced the concept of differential rendering, which provides a framework for rendering the influence of added virtual content on the surrounding real environment. This involves rendering two images, I_R and I_{RV}. I_R consists of a rendering of the real environment model, and I_{RV} consists of the real environment model together with the added virtual content. The difference between these images $D := I_{RV} - I_R$ then provides a good estimate of the influence of the virtual content on the real scene. This can be added to the real input image to produce the final MR output.

Differential rendering is particularly valuable due to its robustness to inaccuracy in the real environment model. Since the rendered real environment model is never observed directly, convincing results can be achieved with even very basic real scene models and approximated material properties. This is very important in AR applications, where often detailed real geometry and material information is not available. This property was demonstrated in [28], where photorealistic results were

achieved despite the real scene model being a simple diffuse plane. It should also be noted that differential rendering is a general framework, and implementations can use any specific GI rendering technique to produce I_R and I_{RV}.

One disadvantage of differential rendering as decribed here is the necessity to render two separate images I_R and I_{RV}. Since both are rendered using a potentially computationally expensive GI rendering approach, producing a single frame of output can be quite expensive. For this reason, some approaches attempt to compute D directly, without separately computing I_R and I_{RV}. Kan and Kaufmann [63] achieved this using a specially adapted ray-tracer, using different categories of ray corresponding to the mixed and virtual scene. In this project, a similar concept is applied to produce D using a single pass of PRT (more detail is given in section 4.3.8).

2.4 Tracking and Reconstruction

A critical feature of MR systems is registration - that the added virtual content responds to changes in the user's viewpoint, in the same way that real content would. MR systems with poor registration tend to have virtual content which does not appear to move together with real content when the user moves their head (in the case of an MR headset) or viewing device (when video see-through MR is implemented on a phone, for example). This can make the virtual content appear less realistic, make it harder for the user to judge spatial relationships between objects and in extreme cases can cause simulator sickness.

Registration requires the current pose of the MR device to be known at each frame, in order to determine how to correctly render the virtual content. This pose is obtained using some form of tracking approach. In the case of video see-through MR approaches using a video camera, The ability to track camera motion and reconstruct the real environment is critical for many mixed reality applications. Tracking methods provide estimates of the pose of the camera, which enable correct registration of virtual content to the real world. Reconstruction approaches can provide important data about the real environment, which can include geometry and colour

information, and in the case of some recent approaches, semantic labelling and material information. These data are also of great importance for rendering realistic virtual content, in addition to enabling interactions between real and virtual objects.

Simultaneous localisation and mapping (SLAM) approaches address the tracking and reconstruction challenges simultaneously. These approaches involve estimating the location of an entity whilst simultaneously constructing a map of its environment. SLAM is a broad field, and has been applied to a wide range of sensing technologies including Radio Detection and Ranging (RADAR), Light Detection and Ranging (LIDAR), Global Navigation Satellite Systems (GNSS) and others. One particular subset of SLAM approaches focuses on imaging devices, including colour and depth cameras. This is referred to here as vision-based SLAM.

Vision-based SLAM approaches can be broadly categorised as dense or sparse. Sparse approaches include MonoSLAM [27], PTAM [79] and ORB-SLAM [101, 102].These construct a map consisting of relatively few, sparsely distributed points. Often, each of these points is associated with an image feature generated using a method such as SIFT [87], enabling them to be relocalised within future images. Some more recent "direct" methods including LSD-SLAM [38] and DSO [37] use per-pixel stereo matching against keyframes rather than keypoint-based approaches, enabling them to capture many more 3D points. For this reason, these approaches are sometimes referred to as "semi-dense".

Dense vision-based SLAM approaches such as DTAM [107] and CNN-SLAM[134] construct a more complete model of the real environment. This model can take a variety of forms - it might consist of a triangle mesh or a grid of voxels, for example. After being exposed sufficiently to the real scene, such approaches have to potential to construct a complete 3D model of the real environment.

Sparse approaches provide camera pose estimations which are very useful for AR applications. They are also frequently less computationally expensive and more robust than dense approaches, in part because implementing loop closure and relocalisation strategies is more straightforward. However, they only provide very limited information about the structure of the real environment. Even semi-dense ap-

proaches can only reconstruct very incomplete models, typically only reconstructing the edges of objects. By contrast, dense approaches can also provide full 3D models of the real environment, enabling a wider range of MR interactions such as differential rendering or physical interactions between real and virtual objects.

It is challenging, however, to capture a dense model using traditional RGB cameras alone, particularly when the environment contains featureless surfaces such as blank walls for which depths are difficult to estimate. For this reason, other sensing devices such as RGBD cameras are often used in dense SLAM methods.

2.4.1 Dense RGBD SLAM

RGBD cameras capture detailed colour and depth information in real time, making them extremely useful as sensing devices for dense vision-based SLAM approaches. In this section, a brief overview of current dense RGBD SLAM approaches is given.

Beginning with the publication of Kinect Fusion [106] and the availability of affordable RGBD sensors such as the Kinect, there has been a rapid increase in the development of GPU-accelerated dense 3D reconstruction methods using these devices. These methods track the motion of the RGBD camera in the world and integrate depth data into a dense 3D geometric model. The majority of recent approaches can roughly be grouped according to the type of world representation they use, into volumetric or surfel-based (sometimes called point-based) approaches.

Volumetric approaches include the original Kinect Fusion approach. These approches store the world representation as samples of an implicit function, typically a variety of signed distance function, on a regular grid. Kinect Fusion used a single fixed grid which only allowed reconstruction within a fixed volume, and was fairly memory inefficient as much of the volume typically consists of empty space. More recent publications have addressed both of these shortcomings. Moving-volume approaches [118, 148] extend the area which can be reconstructed by shifting the reconstructed volume as the sensor moves in the real world, and serialising already captured areas. More recent approaches such as [109, 157, 61, 81] instead address the problem by storing the volume more efficiently, using volume hierarchies or hashing. Dai et al. [26] recently proposed an approach which allows the quality of

previously observed regions to be refined by de-integrating and re-integrating data. All volumetric approaches, however, require significant computation to integrate the input data into the volume, and to render the volume for tracking and visualisation.

Surfel-based approaches such as that of Keller et al. [74] instead use a model consisting of a list of points with associated normals and radii, each defining a small disc which forms part of a real surface. Surfels have long been used as an alternative to connected triangle meshes for representing 3D objects in computer graphics [111]. As compared to volumetric approaches, surfel-based approaches are inherently more memory-efficient as all data stored corresponds to part of a real surface. Creating, updating and rendering surfels is also comparatively computationally inexpensive, as no expensive raycasting operations are required. As compared to volumetric approaches, however, converting the resulting model into a triangle mesh is more complex, as the surfels have no connectivity information. More recent surfel-based dense SLAM approaches have added capabilities such as real-time loop closure [149] and the ability to explicitly detect and handle planar surfaces [121].

Not all RGBD SLAM techniques use volumetric or surfel based representations. Thomas and Sugimoto [137, 138] use a series of bump-mapped planes to store the real geometry more compactly. They later enhanced the representation by using more general parametric surfaces [139]. Salas-Moreno et al. [122] identified objects present in the scene in a database, and then these objects could be represented in the reconstruction using just a database index and 6DoF pose. Kerl et al used a series of RGBD keyframes [75]. Bloesch et al. [10] used a unique representation based on an autoencoder network to store reconstructions in an extremely compact form.

Although the majority of dense SLAM approaches focus on the reconstruction of static scenes, some more recent approaches are capable of overcoming this limitation. Rünz et al. [119, 120] created techniques capable of tracking a number of rigid sub-scenes which can move relative to one another. Other techniques have tackled the problem of reconstructing objects capable of non-rigid deformations

[1, 105, 59]. Dou et al. [31] showed how their non-rigid reconstruction approach could be used to capture performances of actors in real time.

Techniques have also been proposed which use just RGB input, but estimate depth values and apply a dense RGBD SLAM approach to generate a detailed reconstruction of the real environment. These depth values may be inferred via stereo techniques [113], or more recently by using deep learning methods [134].

2.5 Occlusion in Mixed Reality

It can prove challenging to correctly reproduce occlusion in a MR application. Whilst mutual occlusion between virtual content can be efficiently rendered in modern graphics hardware, handling occlusion between virtual and real content is non-trivial. This is because it is a priori unclear where the real content should occlude the virtual content, as the location of the real content is often not known.

It should be noted that there is also a separate problem with occlusion in some optical see-through AR displays. These suffer from the issue that virtual content is unable to fully occlude the real content behind it. This is a hardware limitation of displays such as those in the Microsoft Hololens, and whilst it is an interesting problem, it is not explored here. Here, the focus is on the problem of determining where real content should occlude virtual content, particularly for video see-through applications. A number of authors have attempted to tackle this problem, and some of the suggested approaches are summarised here.

2.5.1 Model-based Approaches

A number of well-established approaches exist for accurately rendering the occlusion of virtual objects by real objects, using a detailed geometric model of the real scene [12]. Provided accurate registration is possible, these approaches can render detailed occlusion between virtual and real objects very efficiently, by making use of GPUs.

Such techniques are only applicable to static or precisely tracked real objects, for which detailed models are available. Consequently, such methods are not capable of resolving occlusion in the common case where the user of an MR application

moves their hand in front of a virtual object. Additionally, should tracking become inaccurate, virtual content will exhibit incorrect occlusion.

2.5.2 Contour Tracking

Another class of approaches attempts to remove the requirement to accurately model the real scene, by tracking the silhouette of the occluding real object(s) [7, 85, 78]. Such methods typically track these object edges using a variant of the active contours (snakes) algorithm [72]. Similar techniques can also be used to refine the occlusion estimates produced by model-based methods [78].

Although such approaches have the advantage of not requiring models of the real scene, they typically require sufficient processing power to track the silhouette contours. The tracked contours may also not be entirely accurate, either due to problems in fitting (e.g. local minima), or to insufficient resolution and/or flexibility of the contour model. The active contours algorithm requires initialisation, which may need to be performed by the user (as in [85]); this may not always be desirable or practical.

These methods also implicitly make the assumption that occluders are opaque objects with continuous, definite boundaries of finite length. This assumption does not hold for a wide variety of real-world occluders. Examples include translucent objects such as smoke or stained glass, as well as objects exhibiting subpixel-scale detail such as hair.

2.5.3 Masks and Mattes

The approaches discussed in sections 2.5.1 and 2.5.2 all attempt to solve the problem of determining where real-virtual occlusions occur in an MR application. Many [142, 67, 7, 85, 78] produce a binary label for each pixel in the augmented image, indicating whether a real-virtual occlusion occurs at that pixel, effectively segmenting the image. The binary image containing these labels, a “mask”, can be used to composite the real and virtual images to produce the augmented image.

This is, however, not a complete solution to the problem, as some pixels in an image may be partially occupied by real content, and partly by occluded virtual

content. Such pixels are common at the boundaries of objects, where their edges bisect pixels, but also occur due to translucent real objects, as well as objects on a subpixel scale (such as hair). Assigning binary labels to such pixels leads to a variety of errors, including aliasing along edges.

For this reason, in the related area of foreground-background separation (e.g. for image compositing), some methods instead attempt to develop a matte containing a scalar 'alpha' value in the range $[0, 1]$ for each pixel [129]. Here, 1 indicates a pixel containing only foreground content, and 0 indicates a pixel containing only background content. Values in the $(0, 1)$ range contain a combination of foreground and background. A high-quality matte can produce better results than a binary mask, avoiding problems such as aliasing along edges ("jaggies"), as well as properly handling small-scale and translucent content [128].

2.5.4 Occlusion and Depth

Where per-pixel depth values are available, these may be used to determine occlusion between virtual and real objects. This may be achieved by simply comparing the depths - where a virtual object has a greater depth than a real object, it can be assumed that the real object occludes the virtual object at this pixel.

Wloka et al. [151] describe an MR system which makes use of this principle to correctly composite real and virtual images. In this application, the real depth measurements are acquired using a stereo camera pair and a stereo matching algorithm. These are then added to the z-buffering process used in rendering the virtual content, preventing virtual objects from being rendered at those pixels where real objects exist at lower depths. This allows for the real and virtual content to be efficiently composited, but may cause aliasing along occlusion edges, due to the lack of sub-pixel occlusion. Kanbara et al. [67] later developed an approach with improved depth estimation, and demonstrated its use on a head-mounted display (HMD).

A number of other papers have focussed on the similar problem of producing an alpha matte, separating a foreground object from its background. Performing this operation on a colour image alone is a challenging problem, and existing solutions typically require user input to specify the approximate object location, and have

a high computational complexity, rendering them unsuitable for use in real-time applications [114, 144].

The additional information provided by a depth map can be exploited to produce mattes efficiently, without requiring user input. Wang et al. [147] made use of depth information to automate the application of two established alpha matting techniques to video sequences. More recently, a number of approaches have been developed to perform video alpha matting in real-time [161, 146, 145]. Applications of such algorithms include inserting participants in a videoconferencing system into a single virtual room, for more natural remote group chat [88]. Crabb et al. [22] proposed a method for real-time matting, based on a cross bilateral filter. Similarly to CVF Occlusion, this approach involves processing the depth and colour images using an edge-preserving filter. However, CVF Occlusion is designed for MR occlusion specifically, and uses a two-step filtering and thresholding approach to provide improved results.

2.5.5 Depth Map Enhancement

As mentioned in section 2.5.4, an accurate real depth map may be used to solve the real-virtual occlusion problem. However, the depth map can only be used to provide binary results at each pixel, which leads to some degree of aliasing along the occlusion edges.

A number of techniques exist to improve the quality of the depth map produced by an RGBD camera, using its colour output. One class of these approaches uses some variety of structure-transferring filter, such as a cross bilateral filter [34], to move detail from the colour image to the depth map, and fill holes. L. Chen et al. [19] presented an approach which used morphological operators and suitably adapted cross bilateral filters to remove incorrect values, fill holes and remove noise from depth maps. A later approach by C. Chen et al. [18] instead finds the improved depth map as the minimum of an energy function using the input depths, estimated confidence values and the location of edges in the colour image.

Many works focus on the situation where a depth map and colour image are available, but the depth map is of lower resolution. Here, the task is to upsample and

refine the depth map, to obtain per-pixel depth values for the larger colour image. Approaches have tackled this problem in a variety of ways, including cost volume filtering [153, 20] and adapted bilateral filters [16]. Du et al. [32] used an edge-snapping approach, and also demonstrated how the output could be used to render improved MR occlusion.

2.5.6 Reconstruction and Occlusion

Ventura & Höllerer [142] demonstrated an MR system which compared a planar reconstruction of the environment to the observed colours to identify dynamic real-virtual occlusions. This approach works well under the assumption that the reconstruction is accurate and the scene is static. However, it implicitly assumes that dynamic objects are always located between the viewer and the virtual object, resulting in incorrect behaviour when (for example) a user places their hand behind a virtual object, or the appearance of the scene changes. The method is also only capable of generating binary segmentations, rather than full alpha mattes.

A number of recent methods have been developed to produce dense, detailed 3D reconstructions of real scenes in real-time (see section2.4.1). This opens up the possibility of using such reconstructions to calculate occlusion, using a method similar to that described in sect. 2.5.1. This allows such techniques to be applied in situations where scene geometry is not known a priori.

Such methods are capable of producing high-quality results, but rely heavily on the quality of the reconstruction, as well as the accuracy of the camera tracking. The quality of the rendered occlusion will be lessened in situations where the reconstruction is not capable of accurately capturing the fine detail or transparency of a real object. Many techniques, including Kinect Fusion and InfiniTAM [61] assume a static scene, and the occlusion will be very inaccurate if this assumption is violated (for example, if a user obscures a virtual object with their hand).

2.6 Real Illumination Capture for Mixed Reality

Existing work on capturing real-world lighting for augmented reality can be roughly grouped into two categories. One uses physical light probes of various forms to

capture the necessary information. The other attempts to recover the lighting by other means. In this section, relevant concepts and existing work in this area are reviewed.

2.6.1 Dynamic Range

Real illumination information for MR applications is typically captured using one or more digital cameras. The images captured by such cameras consist of a regular grid of quantised, usually integer values. Converting these values to obtain the radiance values needed for lighting calculations presents a number of challenges. Firstly, the relationship between pixel value and radiance is dependent upon the properties and settings of the camera, and is often non-linear.

Secondly, since the pixel values are quantised, they can only express radiance values over a limited range. Any radiance below a certain threshold will result in a pixel value of 0, and anything above a threshold will result in the maximum integer value (often 255). This means that the radiance at these pixels is unknown, and thus the dynamic range that can be expressed in such an image is limited. In this thesis, we refer to pixels with a value of 0 as "zero-valued", and pixels with the maximum integer value as "saturated".

Real scenes often have wide dynamic ranges which are difficult to capture within a single image, without incurring significant quantisation error or exceeding the physical capabilities of the camera's image sensor. For this reason, it is often necessary to capture multiple exposures or use dedicated HDR imaging devices in order to capture the full dynamic range of the real scene.

Mann and Pickard [89] developed a method for capturing a series of images with different exposure settings, and merging these into a HDR radiance map. Their technique was further developed and improved by Debevec and Malik [29]. First, a mathematical model called a response curve, describing the mapping from pixel values to radiance for the camera in question is estimated. Then, to capture a radiance map, the camera is mounted on a fixed tripod, and a number of images are captured from the same viewpoint using a number of different exposure times. These can then be converted to radiance values using the estimated response curve. Each pixel

of the output radiance map is a weighted sum of these radiance values. In [29], the weighting uses a score function favouring pixels which have an LDR pixel value in the centre of the range (i.e. 128 for a typical 8-bit image).

The process of capturing a series of similar images using different exposure settings is often referred to as exposure bracketing. The term originated in film photography, where bracketing was used to increase the chances that a well-exposed image would be captured when the photographer was unsure of the optimal exposure settings. The same term is now used to refer to the similar process of capturing images to produce a HDR radiance map.

The radiance map typically consists of floating-point values, or of values quantised using a higher bitdepth in order to store the required dynamic range. Radiance maps cannot be shown directly on conventional LDR display devices, which typically require 8-bit RGB input. Generally, a mathematical function is selected and applied to transform the HDR values to the 0-255 range required for display, in a process known as tone-mapping.

Tone-mapping, or tone reproduction is a problem which has been studied for a long time, dating back to techniques used in film photography. A wide range of tone-mapping techniques exist, but can be broadly categorised into two categories. Global techniques apply the same mapping function independently at each pixel, whilst adaptive techniques modify the function across the image, which can help to preserve detail in both bright and dark regions of the image.

Recently, display devices with higher dynamic range accepting higher bitdepth input (e.g. 10-bit) are becoming more widely available. The dynamic range of these displays is still currently much more limited than that present in real scenes, however, and tone-mapping is still necessary.

Although these approaches ([89, 29]) work well for capturing images of static scenes, artefacts are produced if the camera moves or the scene changes whilst the exposures are captured. This means they cannot be used to capture moving scenes, and are not suitable for capturing HDR video footage. Scene objects which move relative to the camera whilst the the bracketed exposures are captured tend

to introduce so-called "ghosting" artefacts, where multiple translucent copies of the moving object are overlaid, giving a ghostly appearance. More recent techniques [103, 11, 90] attempt to remove such artefacts by modelling and compensating for the motion of the scene.

Kang et al. [69] developed an approach for capturing HDR video footage. Their method involves capturing bracketed exposures at video framerate, matching pixels between neighbouring frames using optical flow, and warping and merging the frames to produce an estimated radiance map corresponding to each captured image. This method requires only a single video camera, but the optical flow calculation and warping are expensive and can fail to produce accurate results in some cases.

Other HDR video techniques focus on using imaging hardware to solve the problem of capturing multiple exposures simultaneously. Several authors [2, 3, 143, 140] introduce approaches using multiple sensors equipped with different neutral density filters and a beam splitter to capture multiple exposures at the same time at video framerate. These can then be merged to produce the output HDR video. The use of ND filters means that all sensors can use the same exposure time, which avoids potential issues with different sensors having different levels of motion blur when capturing moving scenes. Nayar and Mitsunaga [103] achieve a similar result using a single sensor, instead placing small ND filters over individual pixels to achieve spatially varying exposure over the sensor. This removes the need for multiple sensors and a beam splitter, but requires an interpolation process to estimate a HDR image at the resolution of the sensor.

Unger and Gustavson [141] capture and merge several exposures taken using an adapted rolling shutter image sensor. A typical rolling shutter sensor exposes and reads pixels in rows, progressing vertically down the sensor until the full image has been captured. In [141], each row instead captures each of the desired exposures sequentially, meaning that all of the exposures are captured in an interleaved fashion over a very short period of time. Since the captured exposures are so close temporally, they can be merged without introducing much ghosting. However, un-

like the ND-filter based approaches, the different exposures use different exposure times, which can introduce some artefacts due to differing levels of motion blur.

Deep learning approaches have also been applied to estimate HDR images from LDR input. Endo et al. [36] generate high and low exposure versions of a medium-exposure input using two separate CNNs, and composite these to form the final output. Eilertsen et al. [33] instead directly recover the HDR image using a single U-Net style network. Although both approaches are intended to operate on single static images, future approaches tailored for LDR-HDR video conversion may soon be developed.

2.6.2 Light Probes for Mixed Reality

Environment maps have long been used to simulate reflection from virtual objects [9]. They have seen widespread use, particularly as an efficient alternative to ray-tracing for simulating effects such as refraction and mirror reflection [45].

Physical light probes are often used to obtain such environment maps. These can be classified into passive probes, such as chrome spheres, and active probes, such as fisheye cameras. Chrome spheres [54, 28, 44, 4, 68] can provide detailed reflection information, which is useful for rendering specular and translucent virtual objects. Other types of passive probes also exist, however, including diffuse spheres [5], which provide useful information for rendering Lambertian virtual objects. Calian et al. [15] presented a novel probe design intended for directly capturing spherical harmonics, for adding virtual content shaded using Precomputed Radiance Transfer (PRT) [127]. Yao et al. [154] demonstrated a method for using arbitrary objects of known geometry as passive light probes.

Cameras with a wide field of view can be used as active probes [65, 63]. They can also provide detailed, high-frequency lighting information, and, unlike passive probes, do not need to remain in the camera's field of view. Grosch et al. [46] made use of this, placing a fisheye camera at the entrance to a room to measure incoming light, and then simulating light transport within the room to add virtual content. Rohmer et al. [116] demonstrated a technique combining information from multiple fisheye cameras to collect detailed lighting over a volume. Other active

light probes have also been developed, such as that of Matsuoka et al. [94], who used a hemispherical array of photodiodes to find the dominant light direction.

2.6.3 MR Lighting Without Probes

A wide variety of approaches exist that attempt to recover lighting information for mixed reality applications without the use of physical light probes. Such methods typically attempt to recover some information about the light in the real environment indirectly from its effect on the image to be augmented. These approaches can be categorised by the type of lighting model they attempt to fit to the visible scene.

One class of approach fits a point-source model. In this case, the application's goal is to determine the locations and intensities of one or more point sources in the real scene. For example, a number of techniques [123, 124, 125, 51, 6] attempt to identify light source directions by finding shadow boundaries in the real image, and inferring the locations of the point sources casting them. Zheng et al. [160] uses similar techniques to allow real objects to be manipulated in a static image, and to allow their shadows to change consistently.

Frahm et al. [40] present an alternative approach, using a hardware setup somewhat similar to the one proposed here. They combine a television camera with an upward-facing fisheye camera. In a precapture stage, the pair is moved through the scene, and the fisheye camera is used to estimate the position of light sources in the studio. These are then used for lighting virtual content via shadow mapping.

Another possible model type represents the incident light at the point of interest as a linear combination of spherical basis functions, with the Spherical Harmonic (SH) basis being commonly used. These representations are typically appropriate for representing low-frequency lighting. Okabe et al. [110] introduced an inverse illumination technique which also analysed shadows in order to compute a SH representation of the low-frequency illumination in the scene.

Karsch et al. [70, 71] use a combination of an environment map and visible light sources to estimate illumination from a single image. Since the problem of fitting the environment map is severely under-constrained, samples are selected from a database of environment maps captured from similar real-world environments.

Gruber et al. [49, 48, 50] have presented a number of techniques designed for rendering virtual content with consistent illumination, using only the output of a single RGBD camera. These techniques involve capturing a geometric model of the scene using Kinect Fusion [60, 106]. The environment illumination and albedo of the scene are then estimated using an inverse rendering approach. The lighting is captured in the form of spherical harmonics, which can then be used to illuminate virtual content added to the scene. The approach can produce impressive results, but is limited to recovering low frequency lighting.

Meilland et al. [96] presented an approach for probeless AR. Their approach uses the output of an RGBD camera to construct a dense 3D model of the environment. The auto-exposure feature of the camera is enabled to allow detail to be captured in bright and dark areas of the environment. The resulting HDR model is rendered into cubemaps, and processed to find real light locations to use for rendering shadow maps. More recently, Zhang et al. [158] showed how this kind of approach could be used in an MR application for virtually refurnishing rooms, and Rohmer et al. [117] used a similar HDR model to render photorealistic virtual content on mobile and desktop hardware. These approaches capture rich scene information using just an RGBD sensor, but have the disadvantage of requiring the whole scene to be reconstructed before virtual content can be added. They also are limited in their ability to update the model should the real lighting or geometry change. Finally, they can only be used in settings where the environment can be reconstructed using an RGBD sensor (i.e. small indoor environments).

Kan [64] introduced a method for capturing a HDR environment map by projecting and aligning multiple images taken using a mobile device, to form a single spherical image. They later showed how this precaptured map could be used to illuminate virtual objects [66].

Whelan et al. [150] developed a method for reconstructing a scene and detecting the locations of point light sources, using an RGBD camera. This involves observing the locations of specular highlights over time, and applying geometric reasoning to find the 3D location of the light sources which caused them.

Recently, a number of papers have proposed methods based on deep learning techniques to estimate HDR environment maps from single, limited field-of-view LDR images. These approaches form part of a wider movement to apply such techniques to mixed and augmented reality[83]. Gardner et al. [42] presented a technique to directly estimate HDR environment maps from images of indoor environments. Hold-Geoffrey et al. [55] presented a technique to capture a parametric model in the outdoor setting, and more recently Legendre et al. [84] presented a technique to recover full environment maps in both indoor and outdoor settings. These techniques all have the advantage of requiring only a single RGB camera, however they estimate only a plausible approximation of the real lighting, with no guarantee of correctness.

In summary, the existing literature provides useful ways to estimate low-frequency lighting or point light source locations without the use of physical light probes. This is suitable for rendering diffuse virtual objects, but insufficient to render highly specular reflections. Methods also exist to recover high-frequency information without probes, but these typically require a precapture step and do not update in real time to reflect changes in the lighting environment. More recent deep learning approaches can provide estimated lighting based on a single RGB image, but the estimated lighting may not correspond to the true scene lighting.

Part I

Real-Virtual Occlusion

Chapter 3

Real-Virtual Occlusion in Mixed Reality

The content in this chapter comes from the following paper, which was published in VRST 2017:

WALTON, D.R., STEED, A. Accurate Real-time Occlusion for Mixed Reality. In *23rd ACM Symposium on Virtual Reality Software and Technology, VRST 2017* (2017)

3.1 Introduction

The first part of the EngD project focused on the problem of properly handling occlusion in mixed reality scenes. In addition to increasing the perceived realism of MR scenes, proper handling of occlusion is also critical for proper spatial understanding.

Occlusions in MR can be classified according to the objects involved. Where a virtual object occludes another virtual object, this occlusion can be handled using one of a number of well-established techniques such as z-buffering. Real objects occluding other real objects do not need to be handled by the MR system. Handling cases where real objects occlude virtual ones, or vice-versa, is more challenging. This is because typically detailed geometric information about the real world is not available, and it is unclear where these occlusions occur.

One potential way to address this issue would be to make use of a dense 3D

Figure 3.1: MR scene containing a virtual laptop, using a variety of occlusion handling methods. Left: Scene with no occlusion handling (virtual content always rendered over real content). Middle: Scene using occlusion handling based directly on RGBD depth input. Right: Scene using the occlusion handling method detailed in this chapter.

reconstruction algorithm such as Kinect Fusion [60]. Such approaches, however, have a high computational cost and are unable to capture dynamic objects which move or change in shape. This means, for example, that they would be unable to properly handle occlusion in the common case where a user moves their hand in front of a virtual object.

RGBD cameras provide dense depth information per-frame, which in principle could be used directly to solve the occlusion problem. In practice, however, the output of these cameras is noisy and incomplete, and using them directly gives poor-quality results. The centre image in figure 3.1 shows a typical example.

This project focused on developing an approach to make use of both the depth and colour frames produced by an RGBD camera to provide detailed, high-quality occlusion. A number of approaches were developed and tested, with the most successful being named Cost Volume Filtering Occlusion (CVF Occlusion). This approach made use of the cost volume filtering framework developed by [56]. In addition, a method was developed to quantitatively compare such real-time MR occlusion methods.

3.2 Occlusion Method: Details

3.2.1 Hardware Setup

The hardware system used by this method consists of a single RGBD camera observing a scene containing an MR marker. Here, the ArUco library [43] was used to determine camera location, which tracks by using fiducial markers.

Table 3.1: Pixel category names and associated conditions.

'infront'	On virtual object, real depth $<$ virtual depth
'behind'	On virtual object, real depth $>$ virtual depth.
'process'	On virtual object, real depth unknown or equal to virtual depth
'ignore'	Off virtual object.

Although a marker-based tracking system was used here, alternative marker-less tracking systems such as PTAM [80] could also be used for this purpose. The camera location is only used to correctly position the virtual content in the rendering step. A marker-based system was chosen because stable, external tracking was needed to compare a range of occlusion techniques.

3.2.2 Rendering

The virtual content is rendered, using the transforms obtained from the tracking. In addition to rendering RGB pixels, the depth (i.e. z co-ordinate) of each pixel in camera space is also rendered, producing a virtual depth map. The minimum and maximum depths of each virtual object are recorded.

3.2.3 Pixel Categorisation

The virtual depth map and real depth map are compared, to categorise each pixel in the image. These pixel categories are used as input to the subsequent stages of the approach. The categories employed are listed in table 3.1.

The pixel categories are named according to whether the real scene is in front of or behind the virtual object at a given pixel.

As noted by [108], the style of depth camera used here suffers from lateral noise. This noise causes inaccurate depths at pixels near edges in the depth map. If the pixels were classified exactly as described in table 3.1, this would lead to some pixels erroneously being included in the 'infront' and 'behind' categories.

In order to mitigate this problem, a morphological (erosion) operator was used to remove pixels within a 3-pixel range of the border of the 'infront' and 'behind' categories. These pixels were included in the the 'process' category if occupied by the virtual object, or the 'ignore' category otherwise. As noted in [108], the lateral

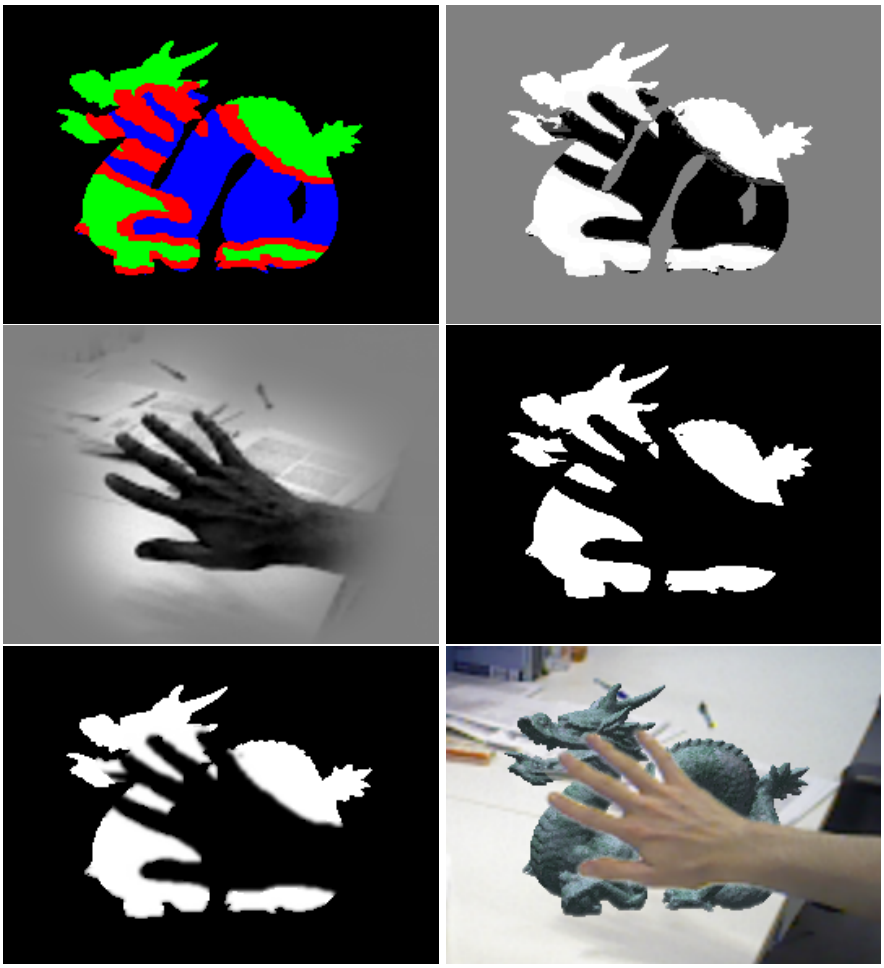


Figure 3.2: Stages involved in generating an augmented image using the proposed method. Top left: Categorisation applied to pixels. Classes are coloured as follows: 'infront', 'behind', 'process', 'ignore'. Top right: Initial costs. Middle left: Filtered costs. Middle right: thresholded costs. Lower left: Final matte. Lower right: Composited MR image.

noise does not vary significantly with depth, so this 3-pixel range could be generally applied.

Note that the pixel categories generated here are similar in concept to the trimaps used in alpha matting techniques such as that of [21]. The difference is that here, in addition to the three typical categories of foreground, background and unknown, we add a fourth 'ignore' category.

The top left image in figure 3.2 shows an example classification of pixels in an to which a virtual dragon is being added.

3.2.4 Initial Cost Calculation

Using these categories, per-pixel costs are generated, using a similar approach to that used by Hosni et al. for interactive image segmentation [56]. In order to calculate these costs, colour models are first fitted to the 'infront' and 'behind' pixel categories, in a small area around the 'process' region. In this implementation, we used histograms, however other colour models such as Mixture of Gaussians could also be used.

Once the models are fitted, the initial costs can then be generated. Each cost is a 32-bit floating-point value, and captures the likelihood of the real object being in front of or behind the virtual object at that pixel.

For pixels in the 'infront', 'behind' or 'ignore' categories, the cost is simply set to 1, 0 or 0.5 respectively. In the 'process' region, it is set to a value in the range $[0,1]$ calculated as follows:

$$C(p) = \frac{P_{infront}(p)}{P_{infront}(p) + P_{behind}(p)}$$

Here, p is the pixel in question, and $P_{infront}$, P_{behind} are the probabilities of the pixel's colour under the respective histograms.

In many cases, these costs already resemble a reasonable image matte. Often, however, it contains inappropriate colour detail. This is removed in the subsequent cost filtering step.

An example of these initial costs can be seen in the top right of figure 3.2.

3.2.5 Cost Volume Filtering

Following this, a guided filter [52] is applied to the costs. The guided filter is an edge-preserving smoothing filter, which takes two images as input; a guidance image and an input image. The filter has the effect of smoothing the input image, whilst preserving hard edges present in the guidance image.

Here, the input image consists of the initial costs, and the guidance image is the colour image from the RGBD camera. The guided filter has the effect of smoothing the costs and removing some of the unnecessary detail, whilst preserving hard edges

in the colour image. These hard edges typically correspond to occlusion boundaries.

This initial cost volume filter uses a fairly broad radius to provide a strong smoothing effect. After the filtering stage, the pixels in the 'process' region typically more closely resemble the desired matte, as can be seen in the middle left image of figure 3.2.

3.2.6 Matte Generation

Following the filtering stage, the filtered costs are thresholded to produce an initial matte for compositing the virtual object. Costs above 0.5 are set to 1.0, and those below 0.5 are set to zero. Only pixels from the 'process' region are used This produces an initial, binary matte.

A second guided filter is then applied to the matte. This filter uses a smaller radius, and has the effect of 'feathering' the occlusion edges. This removes the aliasing artefacts that would result from using the initial binary matte as-is. Results are particularly improved in cases where occluders have 'fuzzy' edges where many pixels should exhibit partial occlusion (for example the hair on a person's head, or a fluffy toy). Examples of thresholded costs and the final generated matte can be seen in the middle right and bottom left of figure 3.2, respectively. Here, one can see that the filter has removed the aliasing artefacts from the edges of the hand.

This process is adapted from the matting approach proposed by [56]. However, Hosni et al. use an iterative approach, where the model fitting, filtering and thresholding stages are repeated multiple times. In our approach these stages are performed only once, improving the efficiency and allowing the process to run in real time. Since the depth map is already close to the true occlusion edges, a single iteration is typically sufficient. Future implementations could use multiple iterations if this is necessary. This would be useful if, for example, the RGBD sensor used has a lower-resolution depth sensor, or higher-resolution colour sensor than the one used here.

If multiple iterations are required, it is possible to take advantage of another property of the guided filter. When multiple images are to be filtered using the same guidance image and filter parameters, much of the computation can be reused to

improve efficiency. This reduces the computational cost of subsequent iterations.

3.2.7 Compositing

Once an alpha matte has been produced, a compositing process is applied to produce the final augmented image.

This process is an approximation on the boundary of the occluding real object (i.e. at pixels where the matte value lies in the range $(0,1)$). Here, the input colours at each pixel c_o initially contain a linear combination of the real background and foreground colours c_b and c_f:

$$c_o = \alpha c_b + (1-\alpha)c_f$$

In principle, the augmented image should have the colour c_a, where:

$$c_a := \alpha c_v + (1-\alpha)c_f$$

Here, c_v is the colour of the virtual object at this pixel. The approaches described here determine α, but not c_f. Consequently, the output image uses c_o in place of c_f. This could produce erroneous composites in certain situations, particularly when the foreground and background colours differ greatly at a pixel exhibiting partial occlusion. We found that this approximation produced reasonable results in our tests, however.

3.3 Implementation & Evaluation Method

3.3.1 Implementation

This implementation of CVF Occlusion used a combination of CPU and GPU processing. The initial pixel categorisation, histogram fitting and cost calculation took place on the CPU, and were implemented in C++. The costs were then transferred to the GPU, and all subsequent processing took place there. This meant that the most expensive component of the approach, the two guided filter applications, could take advantage of efficient GPU implementation. It also minimised the information which needed to be transferred between CPU and GPU at each frame. The GPU

code was written in OpenGL, which was also used to render virtual content and compose the output image. Components such as the guided filter were implemented in compute shaders.

The guided filter implementation used the efficient formulation described in [52]. This consists of a series of box filters, in addition to a number of simple pixel-wise image operations, such as adding, subtracting or multiplying two images. The box filters are implemented using summed area tables [24] (later works often refer to these as integral images). Consequently, our implementation is $O(N)$, where N is the number of pixels in the input image. It is independent of the radius of the filter employed. The summed area tables were calculated using parallel GPU prefix sums along rows and columns of the image.

3.3.2 Quality Evaluation Method

In order to quantify the relative quality of the results, it was necessary to develop a procedure for obtaining ground truth occlusions, and metrics to compare this ground truth to the output of CVF occlusion.

A small environment in which to carry out the experiment was constructed, containing an occluding object and some background objects. The RGBD camera was mounted on the quick-release plate of a sturdy tripod and pointed towards the scene. An MR marker, attached to a stiff foamcore board was placed off to the side of the scene, visible to the camera. The MR scene consisted of a single virtual object, between the real foreground and background objects, arranged to be partly obscured by the foreground.

We also intended to compare the results of CVF occlusion to those of a dense, Kinect Fusion-style algorithm. Such algorithms are intended to construct a scene model of gradually increasing quality over time, and need to view objects from multiple angles to produce accurate models. In order to compare a dense SLAM method to the proposed approach during the reconstruction, the following steps were repeated:

1. Detach the camera from the tripod.

2. Move the camera around the scene for a period of time, allowing Kinect Fusion to reconstruct the scene.

3. Reattach the camera to the tripod.

4. Mark a series of frames for comparison, record the appearance, depth, mask of the virtual object.

5. Move the green screen behind the object, and capture the ground truth frame.

6. Remove the green screen.

The dense SLAM algorithm used for the comparison here was InfiniTAM [61]. This builds upon the original Kinect Fusion approach, adding a volume hashing mechanism enabling it to store the scene model compactly and reconstruct larger areas.

RGBD cameras, such as the one used in this example, an Xtion Pro, typically suffer from some temporal noise, particularly in the depth output. Because of this, it was decided to compare a series of consecutive frames during each cycle, to determine if this temporal noise caused the error of any of the approaches to vary significantly (here, 4 frames were used).

In practice, it was found to be easier to losslessly record the footage from the RGBD camera, and then apply the algorithms afterwards. This approach was chosen as it would not have been possible to run all of the compared approaches simultaneously in real-time.

3.3.2.1 Per-frame Accuracy

The mattes obtained using the occlusion methods were compared to ground truth mattes, in order to assess their accuracy. Figure 3.3 shows the process of developing a ground truth object matte from a green screen image. First, a photo editing tool is used to produce a matte b) from the input green screen image a). In practice it was found that attempting to produce this matte without user intervention sometimes led to inaccurate results for more challenging objects. The pixels of this matte not

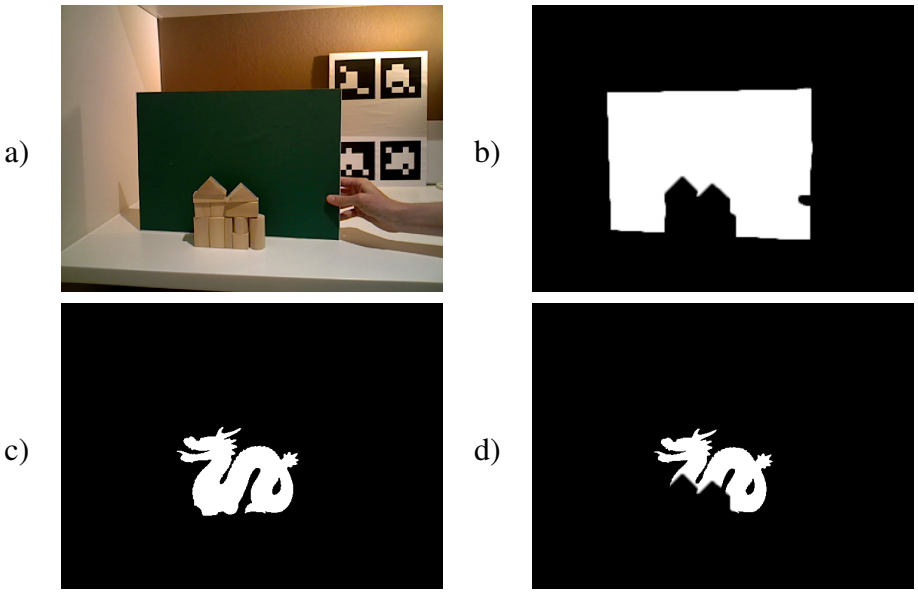


Figure 3.3: Developing a ground truth matte, using a green screen. a): Green screen placed behind foreground object. b): Matte extracted from greenscreen image. c): Pixels occupied by virtual object. d): Ground truth occlusion matte.

occupied by the virtual object c) are then set to zero, resulting in an accurate ground truth matte d).

Once ground truth mattes M_G had been obtained, the accuracy of the automatically generated mattes M was then measured by calculating the mean squared error (MSE), as defined in equation 3.1.

$$MSE(M) := \frac{1}{|\Omega|} \sum_{p \in \Omega} (M(p) - M_G(p))^2 \tag{3.1}$$

3.3.2.2 Temporal Noise

In addition to improving the quality of mattes for individual frames, as measured by the MSE metric, it is also desirable for an occlusion method to remove the temporal noise present in the depth maps. This noise manifests as disturbing flickering artefacts along occlusion boundaries when the depth map is used directly.

In order to assess the ability of the occlusion methods to remove temporal noise, an additional metric was calculated. During step 4 of the capture procedure outlined in sect. 3.3.2, several frames are captured in sequence whilst the camera is fixed to a sturdy tripod. These frames contain static real and virtual scenes and so

ideally the matte output would be constant over each of these sequences.

To measure the temporal noise in a sequence of output mattes, the variance at each pixel location in the matte is computed across the sequence. The mean of these variances is then taken, to provide a scalar measure of the temporal noise. As noted above, the ideal occlusion method would produce a sequence of identical mattes, resulting in a value of zero. A method that produced some degree of temporal noise would produce mattes with more variation, resulting in a higher (worse) value.

There are some limitations to assessing temporal noise in this way, as the camera and real scene are kept static. It is possible that the characteristics of the depth noise vary as the camera moves relative to the scene. Measuring temporal noise when the camera is moving is a more challenging problem, which we leave to future work.

3.3.3 Compared Techniques

This evaluation method was applied to a number of different occlusion methods, to compare their performance.

Two naïve occlusion methods were implemented, which use the depth map from the RGBD camera directly. We refer to these as 'Direct1' and 'Direct2'. 'Direct1' assigns matte values m using just the real, unprocessed depth d_r and the virtual depth d_v at each pixel as follows:

$$m := \begin{cases} 1 & \text{if } d_r > d_v \\ 0 & \text{if } d_r \leq d_v \text{ or } d_r \text{ unknown} \end{cases}$$

'Direct2' is otherwise identical, but sets m to 1 where d_r is unknown. Thus, both methods compare real and virtual depths per-pixel to determine occlusion. In cases where the real depth is unknown, 'Direct1' assumes the virtual object is occluded, and 'Direct2' assumes it is not occluded. Missing values occur on both unoccluded and occluded parts of the virtual object, so neither of these approaches is inherently superior (although one might provide better results in a particular situation).

CVF occlusion was also compared to the earlier method of Crabb et al. [22].

This approach was developed for foreground-background segmentation, rather than MR occlusion. Here it has been adapted by setting the threshold depth to the mean depth of the virtual object, and setting regions of the matte not located on the virtual object to zero. This adapted method is referred to as 'Crabb' in the results.

To provide context, the presented method was also compared to simpler approaches using other structure-transferring filters. In these approaches, an initial matte was generated using the pixel categorisation described in sect. 3.2.3, and the filter was then applied to refine it. The initial mattes were generated by setting pixel values from the behind category to 1, those from the unknown category to 0.5, and those from the infront or ignore categories to 0.

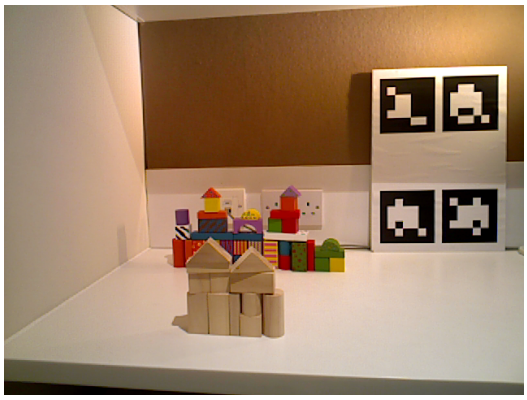
The filters tested were the guided filter and an adapted joint bilateral filter. The baseline guided filter approach uses the same filter implementation as CVF occlusion, but only applies a single filter pass with no thresholding stage. The baseline bilateral approach used a modified joint bilateral filter, adapted to only draw information from pixels in the 'infront' and 'behind' categories. In both cases, the filter was only applied to pixels in the 'process' category. These two methods are referred to as 'Guided' and 'Bilateral' in the results (according to the filter used).

CVF occlusion is referred to as 'CVF' in the results.

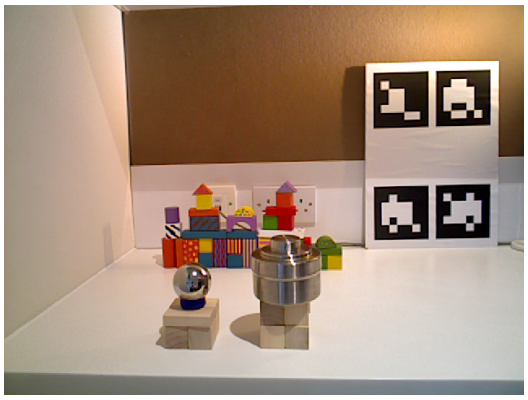
3.4 Results

The experiment was repeated for a number of small MR scenes. These were constructed to be on a scale which allowed the camera to produce accurate depth values, and Kinect Fusion to reconstruct the scene accurately. Images of the scenes constructed are shown in figure 3.4.

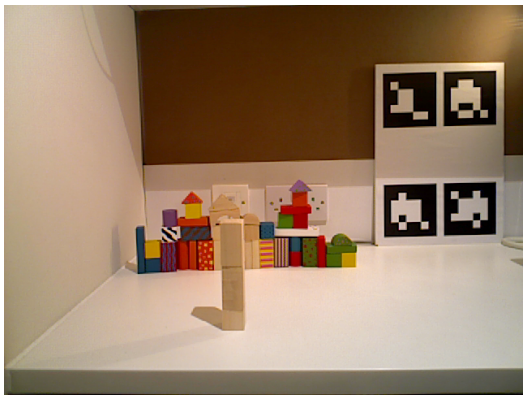
The first scene, 'Normal' was designed to represent a typical use case, and the other two to represent more challenging situations. In 'Same Colour', the foreground and background are of a similar colour near the virtual object. In 'Specular', highly reflective foreground objects mean that the RGBD camera is often unable to obtain accurate foreground depths.



Normal



Specular



Same Colour

Figure 3.4: Colour frames from the three scenes constructed to test the output of the algorithm.

3.4.1 Per-frame Accuracy

Figure 3.5 shows the measured errors of the resulting mattes relative to the ground truth frames, using the MSE error metric. Each bar shows an average MSE for the frames captured consecutively in each cycle. In the first group, InfiniTAM has been exposed to the scene for a short length of time, with a stationary camera. In subsequent groups, the algorithm has been exposed to the scene from a wider range of viewing angles, for a longer length of time. The InfiniTAM error measurements (shown in orange) are the only ones which should show variation over time, as the other approaches do not make use of temporal information. Figure 3.6 shows some example results produced using CVF occlusion, with ground truth for comparison.

In the typical case 'Normal' scene, the proposed method provides better results than both direct approaches. The errors are also below those obtained using the

Figure 3.5: Matte quality for each of the compared frames of the three RGBD sequences, as measured using Mean Square Error.

Figure 3.6: Example results obtained using CVF occlusion on each of the three test sequences (top row), and results using the corresponding ground truth mattes (bottom row).

other per-frame approaches.

Looking at the results of the two direct approaches, one can see that which is superior depends upon the situation. In the 'Normal' and 'Same Colour' sequences, where there are typically more unknown values behind the virtual object, Direct1 has lower errors. However, in the 'Specular' scene where there are many unknown values in front of the virtual object, Direct2 produces better results.

The accuracy of the CVF occlusion is lower in the two more challenging cases. In these cases the MSE is not always lower than both Direct1 and Direct2. However, in the 'Same Colour' example it is better than Direct1, and in the 'Specular' example it is better than Direct2, suggesting that the presented approach is more generally applicable.

The 'Specular' example is an interesting case where the approach of Crabb et al. often outperforms the presented approach in MSE. It should be noted, however, that Crabb suffered from much greater temporal noise in this sequence (see figure 3.7). Whether low temporal noise or MSE accuracy is a more important metric is likely to vary depending upon application domain.

The results from InfiniTAM gradually improve as the scene is exposed to the RGBD camera. For the first few frames in the 'Normal' and 'Same Colour' se-

quences, its error is much higher than the proposed approach. This highlights an advantage of using a per-frame approach; good results can be obtained, even on the first frame of a sequence.

It is also worth noting that, under certain conditions, such as rapid camera movement, reconstruction methods such as InfiniTAM can lose tracking, and this often results in the scene model becoming corrupted. Care was taken to avoid such rapid motions when capturing these sequences. The method presented here does not use any intra-frame state, and thus will continue to operate even when the camera image changes very quickly.

3.4.2 Temporal Noise

Figure 3.7 shows the average temporal noise estimates for each occlusion method, for each captured sequence. The noise estimates were calculated as described in sect. 3.3.2.2 and the average of these values was taken over each sequence.

As can be seen in figure 3.7, for the typical use case example 'Normal', the proposed approach offers a reduction in temporal noise. It is lower than both of the direct approaches, but also lower than Crabb et al., and the simplified guided filter and bilateral approaches. This shows the benefit of the two-step approach and thresholding stage, which reduce the direct dependence of the output matte on the noisy input depths.

In the more challenging 'Same Color' scene, although the MSE was not consistently below that of both direct approaches (as discussed above), the temporal noise of the proposed approach was still lower. In this sequence, CVF occlusion also showed significantly lower MSE than the baseline guided filter approach. The reason for this can be seen in figure 3.8. The thresholding stage used in CVF occlusion tends to produce results with well-defined boundaries, avoiding the errors which can be seen in the guided filter example.

InfiniTAM consistently offered the lowest temporal noise across all scenes, although this comes at the cost of its slow response to changes in real scene geometry (for example, if an occluding real object is moved).

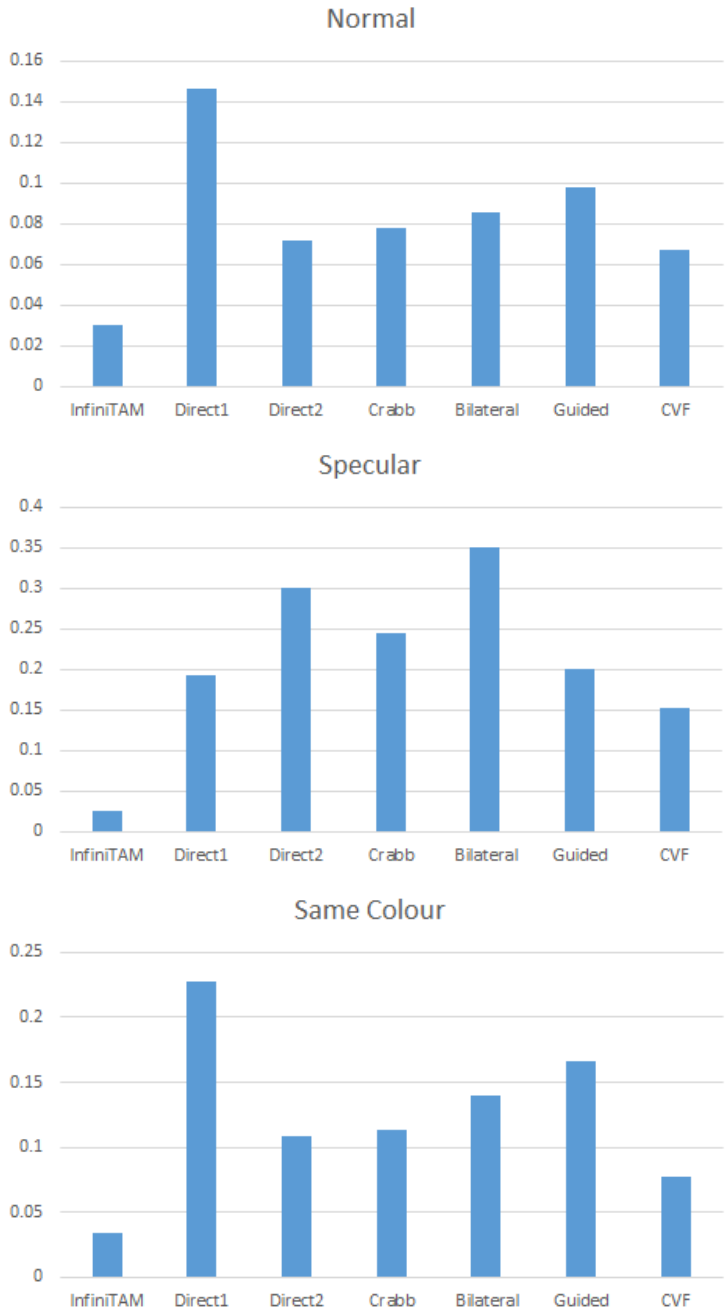


Figure 3.7: Temporal noise for each of the compared frames of the three RGBD sequences, measured as described in sect. 3.3.2.2.

Figure 3.8: Example results from the 'Same Colour' sequence, using the presented approach (left) and the baseline approach using a single guided filter only (right).

3.4.3 Performance

In order to demonstrate the capability of CVF occlusion to work in real-time, a simple MR application was developed. This overlaid a single virtual object on the color image provided by the RGBD camera, using a fiducial marker for tracking. The occlusion method was applied at each frame, and the resulting matte used to compose the final MR image, which was displayed on the screen.

This application was tested using a computer with an Intel Core i7 processor and an Nvidia GTX 1080 GPU. An Asus Xtion Pro depth camera was used for both colour and depth input, at depth and colour resolutions of 640x480 pixels. Using this setup, the application was able to run consistently at 30 frames per second using CVF occlusion, the limiting factor being the framerate of the camera.

The occlusion method was also timed in isolation. In order to obtain meaningful values, it was necessary to forcefully synchronise the GPU (using `glFinish`). This means that the results may not reflect use of the method in a larger application, where the GPU could potentially run other tasks in parallel. They do provide a useful upper bound, however. With this synchronisation, each frame could be processed in between 5 and 7 milliseconds. Due to the $O(N)$ implementation of the guided filter, the processing time was largely independent of the kernel size used, and the proportion of pixels in the 'process' category, offering a consistent framerate.

The implementations of the Bilateral, Guided Filter and Crabb et al. approaches used here were implemented on the CPU, and able to run at near real-time

(40-50ms per frame). More optimised GPU implementations of these approaches would likely have similar performance to CVF occlusion. Thus we believe that the cost of CVF occlusion is comparable with these other approaches, and it may be faster in many use cases.

3.5 Discussion

CVF occlusion implicitly assumes a single virtual object - that is, that all real points lie either in front of or behind the virtual object. The same techniques could be applied to more complex scenes with alternating layers of virtual and real content, by processing each virtual layer separately. It would also be possible to process occlusion for each virtual layer in parallel.

The implementation of CVF occlusion used for the evaluation was not carefully optimised. It would be possible to further improve it to reduce the cost or allow implementation on lower-power mobile platforms. The summed area table calculation used in the guided filter in particular was implemented using two filter passes, and efficiency could be improved using the formulation proposed by [104].

Occasionally, the depth camera will be unable to register any valid depth values for an occluding foreground object, due for example to small size, translucency or specularity. In these cases the object is only visible in the depth image as a patch of unknown values, surrounded by background pixels. In this case, the proposed approach will then incorrectly render the virtual object in front of the occluder. This problem is difficult to address using the information available to the algorithm, but was found to occur relatively rarely in practice.

3.6 Conclusion

This chapter focused on the development of CVF occlusion, a method of producing accurate occlusion mattes from colour and depth input from an RGBD camera. CVF occlusion operates independently on each frame, and is therefore capable of working immediately, and handling dynamic and deformable objects. A novel comparison approach was used to demonstrate that the method offered comparable results to a dense reconstruction approach (InfiniTAM), at lower computational

cost.

This approach could be used directly to address the MR occlusion problem without the need for full 3D scene reconstruction. In future work, however, it would be interesting to explore combining techniques like CVF occlusion with 3D reconstruction methods. This has the potential to combine the best features of both approaches - the temporal stability of 3D reconstruction methods and CVF occlusion's ability to handle dynamic real objects.

Alternatively, future image-based approaches could use information from preceding frames to further reduce the effects of temporal noise in the final output mattes, as well as allowing for better handling of cases where there are no valid depth values for a foreground object in a single frame (as discussed in sect. 3.5).

It would be interesting to explore the possibility of automatically detecting challenging scenarios, and adapting to them. For example, in the 'Same Colour' sequence shown here, it would be possible to note that the background and foreground histograms are very similar, and adapt the subsequent stages of the approach.

This approach currently attempts to solve the occlusion problem by producing a matte, containing alpha values for each pixel. Strictly speaking, as mentioned in sect. 3.2.7, this is insufficient, and compositing also requires the real foreground colour to be extracted. Future methods might attempt to do so, in order to produce more accurate augmented images in the presence of translucency.

CVF occlusion could be a good fit for the GPUs available on mobile devices. A more optimised implementation of this approach could form part of an augmented reality system implemented on a mobile device, such as a Google Tango device, or a phone equipped with a stereo camera pair. It could also be applied on an MR headset, such as the Microsoft Hololens or Magic Leap One.

Part II

Real Illumination Capture

Chapter 4

Probeless Illumination Capture for Mixed Reality

The content in this chapter comes from the following paper, published in ISMAR 2017:

WALTON, D.R., THOMAS, D., STEED, A., SUGIMOTO, A. Synthesis of Environment Maps for Mixed Reality. In *16th International Symposium on Mixed and Augmented Reality, ISMAR 2017* (2017)

4.1 Introduction

Ensuring that rendered virtual objects appear consistent with the real world is an important goal in the field of mixed reality (MR). Part of this involves rendering the virtual objects in such a way that they appear to be illuminated by the world around them, by reproducing effects such as shadowing, reflection and refraction.

Rendering these effects requires information about the lighting environment, which is often captured using light probes. These light probes may be objects with known geometry and material properties, such as chrome or glass spheres. Alternatively, a camera with a suitably wide field of view may be used. When such a probe is placed at the location of the virtual objects, the information can be used to light the virtual objects accurately.

However, there are applications where it may be impractical to place physical light probes in the real scene. In a mobile see-through AR app, for example, it would

be preferable to use a single, self-contained device, such as a phone, tablet or a HMD such as the Hololens or MagicLeap One. A variety of methods exist to capture lighting information without the requirement to place separate physical light probes in the real scene. Such methods, however, typically capture lighting in the form of point light source locations [123, 124, 125, 51, 40, 6] or low-frequency spherical harmonic maps [110, 49, 48, 50]. These are useful for illuminating virtual content with diffuse shading, but, unlike physical light probes, cannot be used to render effects such as mirror reflection and refraction of light. Other approaches [96] capture these higher frequencies, but require the whole environment to be reconstructed as a preprocess, and cannot respond correctly to subsequent changes in the real environment.

The method introduced here attempts to recover full environment maps at the location of each virtual object, which can be used to render these high-frequency effects. The method uses an RGBD camera and a small fisheye camera, contained in the same unit. An indoor, single room environment is assumed, and it is assumed that rough geometry for this room is available (a 2D floorplan, the height of the ceiling, and the initial RGBD camera pose). Both cameras are used to construct and update a detailed model of the room, which can then be used to render the environment maps at any place in the scene in real time. The 3D model of the scene consists of a dense, detailed model of the surfaces around the virtual objects, and a coarse model of the walls, ceiling and floor of the room. Both are updated in real time, enabling the virtual objects to respond to changes in the environment. This division of the real scene into distant and nearby components is similar to that proposed by Debevec [28].

Creating and updating this model is challenging. The main contributions detailed in this chapter are as follows:

1. An efficient method for updating the texture of the coarse model using the fisheye camera, whilst correctly handling occlusions.

2. A method for responding to dramatic lighting changes in the room, updating the whole model including regions not visible to the camera pair.

Figure 4.1: Image augmented with virtual reflective teapot, rendered using the proposed system. Note the reflections of the nearby real objects. No physical light probe was placed near the teapot.

3. A system which uses these data to produce accurate environment maps at desired locations in the real scene.

These contributions were demonstrated by implementing a real-time AR system capable of rendering virtual objects with a variety of material properties in a number of challenging environments. The system was also evaluated, comparing the results produced to those achievable using the fisheye camera directly. Overall, the results suggest that our approach can produce more accurate results and reduce visual artefacts.

4.2 System Overview

This section contains an overview of the structure of the system, and details of the hardware setup used.

4.2.1 Data Flow

Figure 4.2 contains a flowchart, giving an overview of the presented system. The inputs to the system (shown in pink) consist of RGB and depth frames from the RGBD camera, fisheye images from the upward-facing camera and rough geometry for the room (generated from the floorplan as described in section 4.3.1). The method produces as output an environment map, rendered from the virtual object's

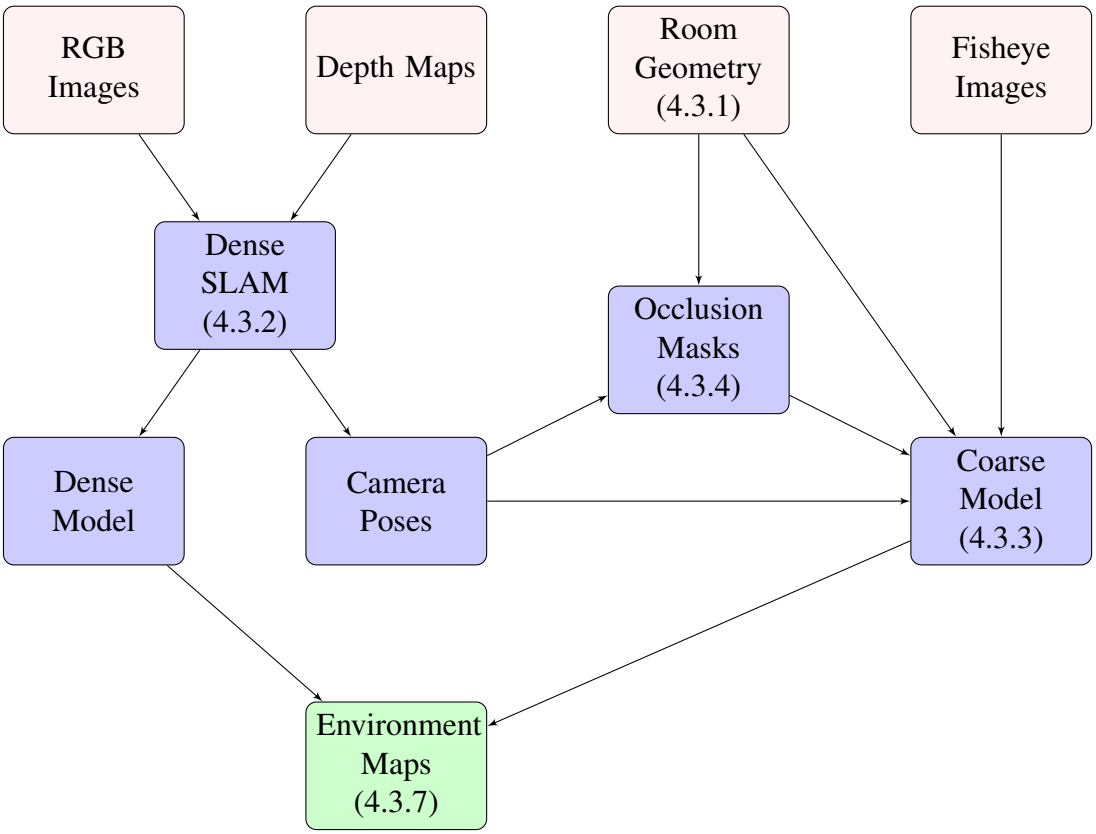


Figure 4.2: System overview flowchart. Inputs are shown in pink, intermediate data in blue, and the output is shown in green.

location. This environment map can then be used to render one or more virtual objects, as described in section 4.3.8.

The coarse model geometry is generated as a preprocess, as the application starts. The other stages are performed in real time, as new frames arrive from the RGBD and fisheye cameras. A new environment map is rendered at each frame, prior to rendering the virtual object.

4.2.2 Hardware Setup

The hardware setup used in this system consisted of an Xtion Pro Live RGBD camera attached to a small upward-facing camera by a rigid bracket (Point Grey Chameleon3 CM3-U3-13Y3C-CS). The upward-facing camera is equipped with a 182 degree fisheye lens (Lensagon CF5M1414), as shown in figure 4.3. The device is intended to be held by the user, with the RGBD camera facing forward, and the fisheye camera facing upward. The colour output of the RGBD camera is displayed

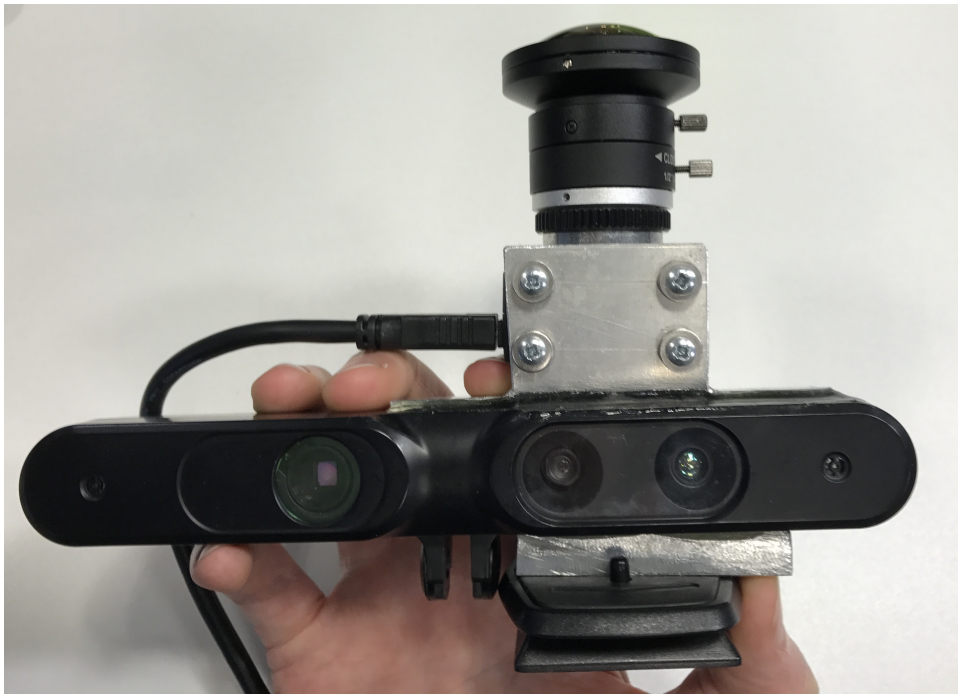


Figure 4.3: Hardware setup used for the proposed method.

to the user, augmented with the virtual content.

Both of the cameras were calibrated. This involved calibrating the cameras individually, in addition to finding the 6DoF transform between the two cameras.

The RGBD camera was calibrated using the camera calibration app from MATLAB's Computer Vision Toolbox [93], which was applied to the RGB images. The camera used for this implementation was capable of automatically warping the depth image to correspond to the RGB image, and this feature was activated.

The fisheye camera was somewhat more complex to calibrate. There are a range of camera models available for omnidirectional cameras, and each model has advantages and disadvantages, in terms of accuracy and computational complexity. Here, the camera was first calibrated using the calibration method of Scaramuzza et al. [126]. This provides a high-accuracy camera model, but it is also unfortunately somewhat expensive to evaluate. For this reason, at runtime, the images were processed so that they conformed to the simpler model of Ying and Hu [155]. This undistortion stage was performed efficiently using a precalculated look-up table (LUT).

Once both cameras were calibrated, the 6DoF transform connecting their camera poses was recovered. This was achieved by finding point matches in the overlapping region in the images captured by the two cameras, and solving for the transform which best aligned the matches, using the Levenberg-Marquardt algorithm [86, 91].

For this implementation, a single upward-facing fisheye camera was used, capturing roughly the hemisphere above the camera pair. We chose this configuration as it was capable of capturing much of the scene that would be reflected by a virtual object, including most major light sources. It would also be possible to use the proposed approach with other camera configurations, for example adding a second fisheye camera to capture a full 360 image. In this case, however, if the device is held by a user, they will be visible to the camera pair, and this may need to be handled explicitly in the coarse model updating stage (section 4.3.3).

4.3 System Detail

This section covers each of the stages involved in the system in greater detail.

4.3.1 Initialising the Coarse Model

The output of the fisheye camera cannot be used directly as an environment map. This is due to the displacement between the camera and the virtual objects, as well as its limited field of view - the camera used here captured a roughly hemispherical region. One of the main goals of the software is to interpret the output of both cameras, to synthesise an environment map at the location of the virtual object.

In order to render the virtual environment maps, a geometric model of the scene is created. This consists of two components. The first is a dense 3D model generated using the RGBD data, capturing the region directly around the virtual objects. The second is a less-detailed, coarse model of the whole room. This is generated from the rough geometry of the room, which is assumed to be known a priori; the floorplan of the room (a 2D polygon) and the height of the ceiling. From this information, the vertices, indices and texture coordinates of the coarse model are generated automatically.

It is assumed that the system is used indoors, within a room. This was felt to be

a reasonable assumption, as this corresponds to the typical use case for the system, and it is not generally possible to use current RGBD cameras outdoors. For outdoor scenes, other environment models could be developed.

For the examples shown here, in order to simplify the implementation, the floorplan and ceiling height were measured by hand. In an end-user application, however, this information could be obtained using the output of the camera pair, for example by applying the method of Cabral and Furukawa [14] to a suitable image captured by the fisheye camera. Alternatively, a SLAM technique such as LSD-SLAM could be employed. Whilst 3D reconstruction using the depth camera would also be possible, it would be more time consuming due to the narrow field of view of the depth sensor, and in practice was found to be challenging due to loss of tracking on flat featureless walls.

4.3.2 Dense SLAM

The dense SLAM stage takes RGB and depth frames as input, and uses these to construct a detailed colour model of the region around the virtual content. Additionally, the SLAM process provides an estimated transform for the RGBD camera. Using the known transform between the cameras (see section 4.2.2), the pose of the fisheye camera can also be estimated.

The dense SLAM algorithm used in this implementation was InfiniTAM [61]. InfiniTAM was selected primarily for its efficiency, but other RGBD reconstruction methods could also be used, such as ElasticFusion [149] or the parametric surface approach of Thomas and Sugimoto [139].

4.3.3 Updating the Coarse Model Texture

As each new frame is captured by the fisheye camera, it is used to update the texture of the coarse model of the whole room. This is achieved efficiently by making use of graphics hardware. Updating the coarse model frequently enables the model to capture changes in the distant real environment in real time, such as a video playing on a TV screen, or curtains being opened.

First, the current location of the fisheye camera is determined using the position

of the RGBD camera, and the known transform between the two cameras. The RGBD and fisheye cameras are not synchronised, however, and may capture frames at different times. We correct for this, estimating the RGBD pose at the instant the fisheye frame was captured, and using this pose estimate to find the correct fisheye pose. This is achieved by linearly interpolating, using the two most recent RGBD poses, and the times at which the images were captured. The rotational components of the transforms are interpolated by converting to quaternion form and applying spherical linear interpolation.

Second, a fragment shader is applied to each texel of the coarse room texture. This identifies the position in world space that the texel corresponds to, projects it into the fisheye camera image, and, providing the image location is valid (i.e. the point is not behind the camera), samples the fisheye image and renders the result to the texel.

This procedure is efficient, but does not account for the possibility of occlusions, in the event that the coarse room model is not convex (the middle left image in figure 4.5 shows an example of such a non-convex floorplan). The process used to handle occlusion is detailed in the following section.

4.3.4 Handling Occlusion in the Coarse Model

Determining which parts of the coarse model texture are currently visible to the fisheye camera is potentially a challenging problem to solve in real time. A naïve approach might involve casting a ray from the camera location to each texel, but due to the large number of texels to be processed this would be prohibitively slow. However, we can exploit the structure of the model to simplify this task. Since the model is a right prism, it suffices to determine which parts of the 2D floorplan are occluded.

Finding which parts of a polygon are visible from a given viewpoint is a well-studied problem in the literature, and very efficient approaches are available. For this implementation, the implementation of Bungiu et al. [13] from the CGAL library [136] was used. The visible region is referred to as a visibility polygon.

Once the polygon has been determined, it is then used to generate a mesh in the

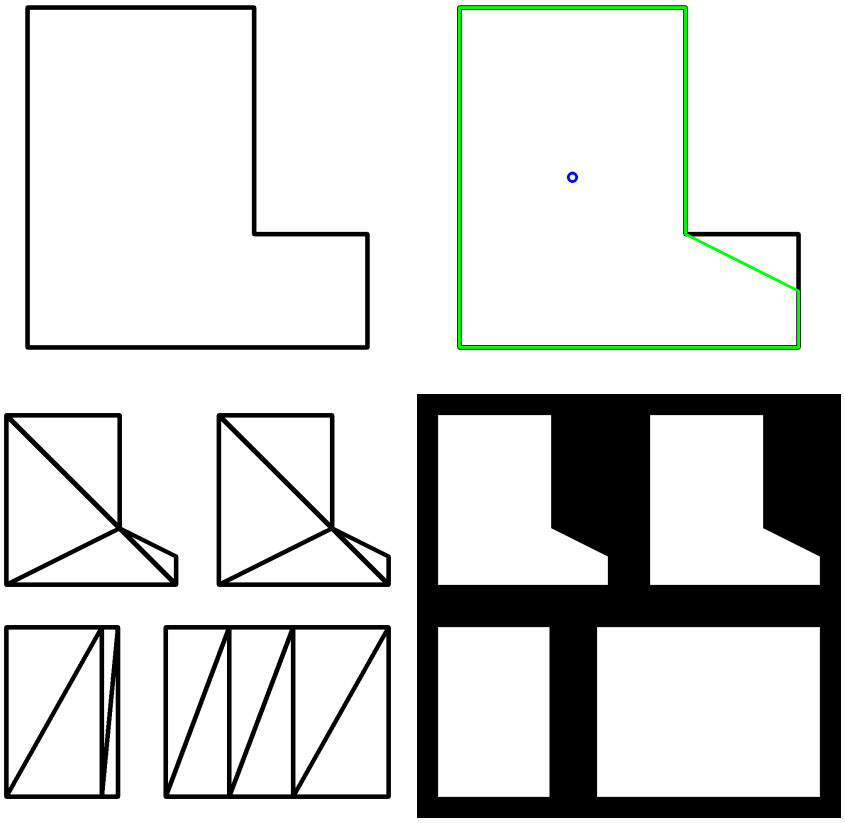


Figure 4.4: An example of generating an occlusion mask for the room model. Top left: floorplan of room model. Top Right: a viewpoint (blue circle) and the associated visibility polygon (in green). Bottom left: mesh rendered into occlusion mask. Bottom right: final occlusion mask.

texture space of the coarse model, which is rendered using the graphics hardware to generate a binary occlusion mask efficiently. This mask indicates which texels are potentially visible to the fisheye camera, and is used during the coarse model update step (section 4.3.3). An example of the generated texture-space mesh and occlusion mask for a hypothetical L-shaped room can be seen in figure 4.4.

Figure 4.5 shows a real-world example. Here, the real environment consists of two rooms, partly separated by a partition, as can be seen in the panoramic image and floorplan. The example is taken from the first frame of the sequence. A 2D visibility polygon is computed from the floorplan, based on the fisheye camera's location (shown as a blue frustum). This is then converted into the binary occlusion mask for the 3D coarse room model. Finally, the coarse model texture is updated with the reprojected fisheye camera image. Only visible components are updated -

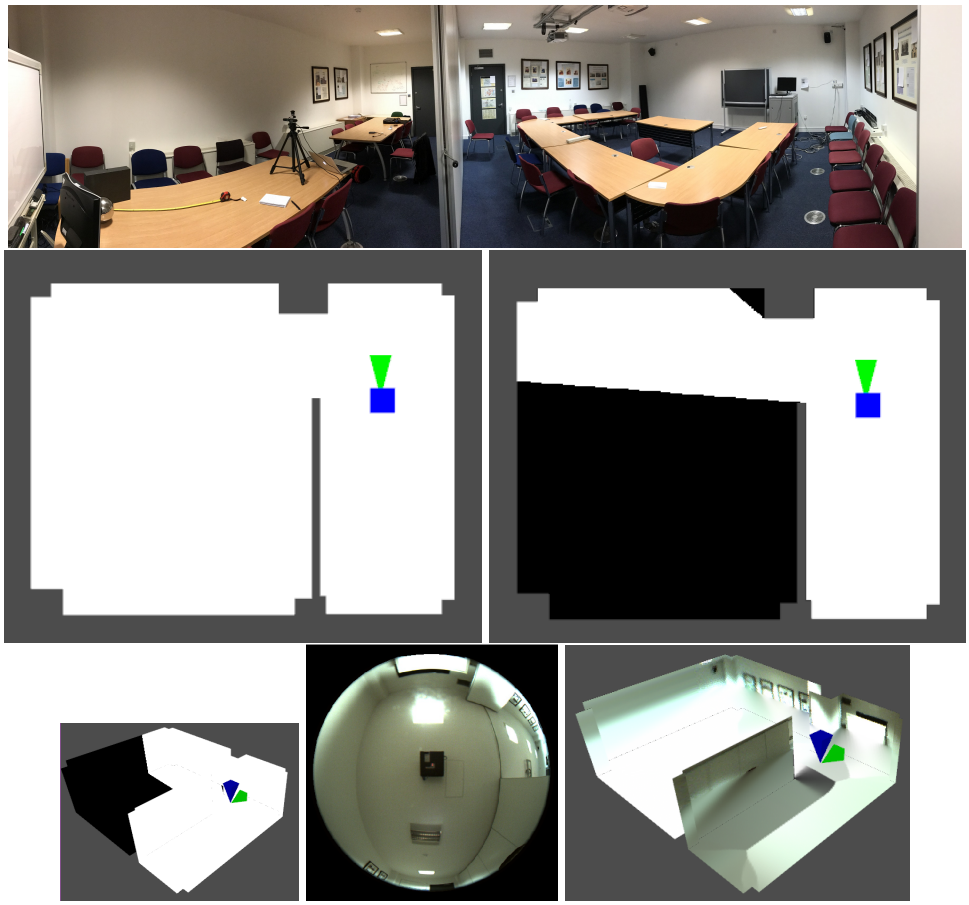


Figure 4.5: An example of occlusion handling when updating the coarse model. Top: Panoramic image of two rooms, partly separated by a partition (centre of image). Middle left: floorplan of rooms and partition. Middle right: 2D visibility polygon. Bottom Left: 3D coarse model, textured with visibility mask. Bottom middle: Frame from fisheye camera (undistorted) Bottom right: Coarse model, updated with this fisheye camera image. In all examples, the poses of the fisheye and RGBD cameras are indicated by blue and green frustra, respectively.

the rest of the texture is filled in using the inpainting approach described in section 4.3.5.

In many cases, the coarse model is in fact convex (i.e. the floorplan is a convex polygon). In these cases, this occlusion testing step is not necessary, as the coarse model cannot occlude itself, and it can be skipped for added efficiency.

4.3.5 Completing Missing Regions of the Texture

At a given time, some parts of the coarse model texture may not yet have been observed by the fisheye camera. This may be due to the camera only having seen the upper hemisphere of the room, or also due to occlusions, if the coarse model is not

convex. In order to complete the missing regions of the texture, an inpainting approach [135] is applied. One advantage of this approach is its speed, which enables it to be reapplied incrementally at runtime as more of the room is observed, keeping the room texture consistent. A binary inpainting mask is used to indicate which texels have been updated with real data, ensuring that the inpainting is only applied to areas which have never been observed.

Inpainting is generally applied to the walls, ceiling and floor separately, as they are typically of different colours. An exception to this is made in situations where the floor has not yet been observed by the fisheye camera. In this case, the inpainting process propagates data to the floor from the surrounding walls to provide a plausible initial texture.

Figure 4.6 demonstrates the effect of this inpainting process on the first frame of a sequence. Here, in the example without inpainting, much of the coarse model has not yet been observed. The texture was initialised to a neutral grey colour, which is visible in the reflection from the virtual teapot. In the example with inpainting, the walls and floor have been inpainted, resulting in a more plausible reflection from the virtual teapot.

4.3.6 Reacting to Large-Scale Lighting Changes

Changes in the lighting environment, such as lights turning on or off, or curtains being opened, have a global effect on the appearance of the room. The fisheye camera is able to observe a large portion of the room, and can update areas in its field of view in real time. Regions not currently visible to the camera are not updated by the system described above, however. If the room becomes dramatically brighter or darker, this can lead to noticeable artefacts. These typically take the form of sudden lighting changes across the coarse model texture, as shown in figure 4.7.

An efficient method was developed to address these problems. At each frame, before integrating the fisheye frame into the coarse model texture, a global illumination change Δ is estimated. This RGB colour value is an estimate of the average intensity change over the coarse model texture, relative to the previous frame. This is computed as part of the coarse model update step.

Figure 4.6: Scene containing a virtual teapot, with and without inpainting applied. Top: augmented image (with inpainting). Middle: view of coarse model and closeup of teapot, without inpainting. Bottom: view of coarse model and closeup of teapot (with inpainting).

For each coarse model texel to be updated with new data from the fisheye camera, the intensity difference between the current and new pixel values is calculated. The mean of these differences Δ is then found. When calculating the mean, parts of the coarse model which have not yet been updated with real data are excluded. These areas will contain inpainted texels, which do not necessarily reflect the true appearance of the room. To identify these pixels, the binary inpainting mask is used (see section 4.3.5). Saturated and black pixels are also avoided, as their true brightness is unknown. More precisely:

Figure 4.7: Example of adjusting to a change in lighting conditions. Above: Image with ceiling light turned off. Middle: Image after ceiling light has been turned on, without lighting change estimation, and closeup of virtual sphere. Below: Image after ceiling light has been turned on, with lighting change estimation, and closeup of virtual sphere.

$$\Delta = \frac{\sum_{t \in T} v(t) \cdot (F(f(t)) - T(t))}{\sum_{t \in T} v(t)}$$

Here, T is the coarse model texture, and t a texel location. F is the fisheye image, and f is a function taking texel locations in the coarse model to the corresponding pixel locations in the fisheye image. v is a validity delta function, defined as follows:

$$v(t) = \begin{cases} 1, & \text{if } t \text{ is visible to fisheye camera} \\ & \text{and } T(t) \text{ is not saturated/black} \\ & \text{and } F(f(t)) \text{ is not saturated/black} \\ & \text{and } t \text{ has been directly observed} \\ 0, & \text{otherwise} \end{cases}$$

This change Δ is added to all texels in the coarse model which were not updated with new data from the fisheye camera. This has the effect of propagating changes in lighting to these texels, ensuring the coarse model appears to be of a consistent brightness. Δ is also used to adjust the appearance of the dense model when rendering the cubemap.

An example is shown in figure 4.7. Here, a ceiling light was turned on, increasing the brightness of the room. In the middle example, without lighting change estimation, the lower half of the reflected scene (i.e. the sewing machine, table and box) has not changed, and now appears too dark. In the lower example, the change in illumination was accounted for and the brightness and colour of the lower half of the sphere appear more consistent.

This approach uses a simplified ambient lighting model. In reality, light transport through the scene is much more complex. This approximation is efficient, however, and helped to reduce visible brightness inconsistencies in the environment maps.

4.3.7 Rendering the Environment Maps

The virtual environment maps are then rendered, using the coarse and dense models. One environment map is rendered from the perspective of each virtual object. Each environment map is rendered in the form of a cube map, which can then be used directly by the graphics hardware.

First, the textured coarse model is efficiently rendered to the cubemap, using the graphics hardware (each face of the cube is rendered in turn). Secondly, the dense model captured using InfiniTAM is rendered. The dense model is rendered on top of the coarse model (i.e. without depth testing).

This approach was chosen because the dense model is typically closer to the virtual object than the coarse model. Additionally, in cases where the dense SLAM captures geometry already present in the coarse model (i.e. a wall, ceiling or floor) this ensures the version from the dense model is visible. This provides improved results, as the dense model typically has more geometric and texture detail. The dense model is rendered using the GPU-based raycasting procedures in InfiniTAM.

Although the examples shown here involve a single virtual object, if others were present, they could also be rendered into the cubemap at this stage, allowing the environment map to capture the complete mixed reality scene.

4.3.8 Rendering the Virtual Objects

Environment maps provide much richer information about the lighting environment around a virtual object than simpler real-time lighting models such as point and directional lights. Although originally developed to produce mirror reflection [45], they can be used to simulate a wide variety of virtual materials. Some examples were implemented below, to demonstrate the capabilities of the proposed system.

Environment maps can be used to produce convincing refraction effects, an example of which is shown in figure 4.8. The top example shows simulation of refraction through a bottle of water. The lower two examples show an example of rendering a metallic teapot, simulating material colour and surface roughness. Surface roughness can be simulated by either combining suitable environment map samples, or by applying a filter to the environment map [100] (or both [82]). The ex-

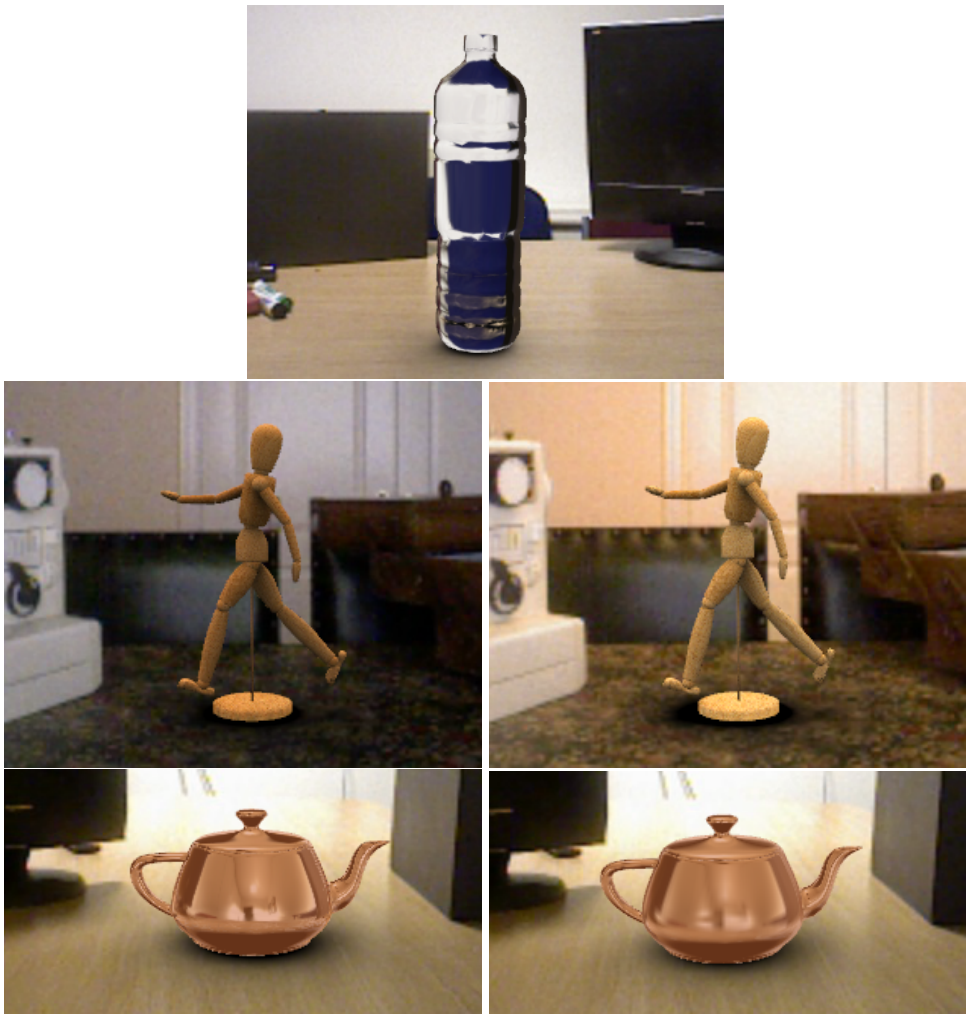


Figure 4.8: Examples of some of the other virtual objects that can be simulated using environment maps. Top: Water bottle, simulating refraction of the environment. Middle: Wooden mannequin, simulating diffuse shadowing, reacts to a ceiling light being turned on. Bottom: Metallic teapots, simulating different levels of surface roughness.

amples shown here simply sample from coarser mipmap levels, which has a similar effect.

Environment maps are not limited to rendering specular virtual objects. They can also be used to render diffuse objects - for example by approximating the environment map using directional lights, via importance sampling. They can also be projected into an SH representation, and used as input to PRT rendering [127], as shown in figure 4.8, middle. In all the examples shown here, diffuse shadowed PRT is used, with a 4^{th} order SH projection (i.e. 25 coefficients).

The implementation of PRT used here also renders a contact shadow onto the table, using differential rendering [28]. Rough real geometry is required for this when precomputing. Here a plane is used, implicitly assuming that the virtual object is placed on top of a locally planar real surface at runtime. This is a common scenario, as realistic virtual objects are typically placed on tables, floors etc.. This allows the virtual object to cast a convincing contact shadow. Since the cubemap is updated and reprojected anew each frame, the virtual object also responds in real time to lighting changes, such as the ceiling light turning on in figure 4.8 above.

Figure 4.9 shows the process of calculating the differential PRT textures for a virtual object (in this case, the Spot model created by Keenan Crane). A PRT texture is baked for the virtual object on the plane, resulting in the texture on the bottom left. A texture is then generated for the real plane alone, giving the result shown in the middle. The final output is the difference of these two textures, shown on the bottom right. Note that in this case, the PRT values on the plane in the lower half of the texture are negative-valued - this will produce a contact shadow when the texture is used for rendering.

4.4 Results

This section contains results of using the system in a typical setting. Qualitative comparisons with a real reflective object and a simpler baseline method are shown.

4.4.1 Full Pipeline Results

Figure 4.10 shows an example of the data captured when using the application in a typical setting. The top and middle images show views of the textured coarse model, and the estimated camera locations. The lower image shows a raycast of the dense model, from the viewpoint of the RGBD camera. The appearance of objects near the virtual object (here, the sewing machine, table and box) are captured in the dense model. The appearance of more distant parts of the environment (e.g. walls, ceiling) are captured in the coarse model. Note that these data are from the end of the first frame of a run using the application, so some quantisation noise remains in the dense model, and the lower half of the coarse model mainly contains inpainted

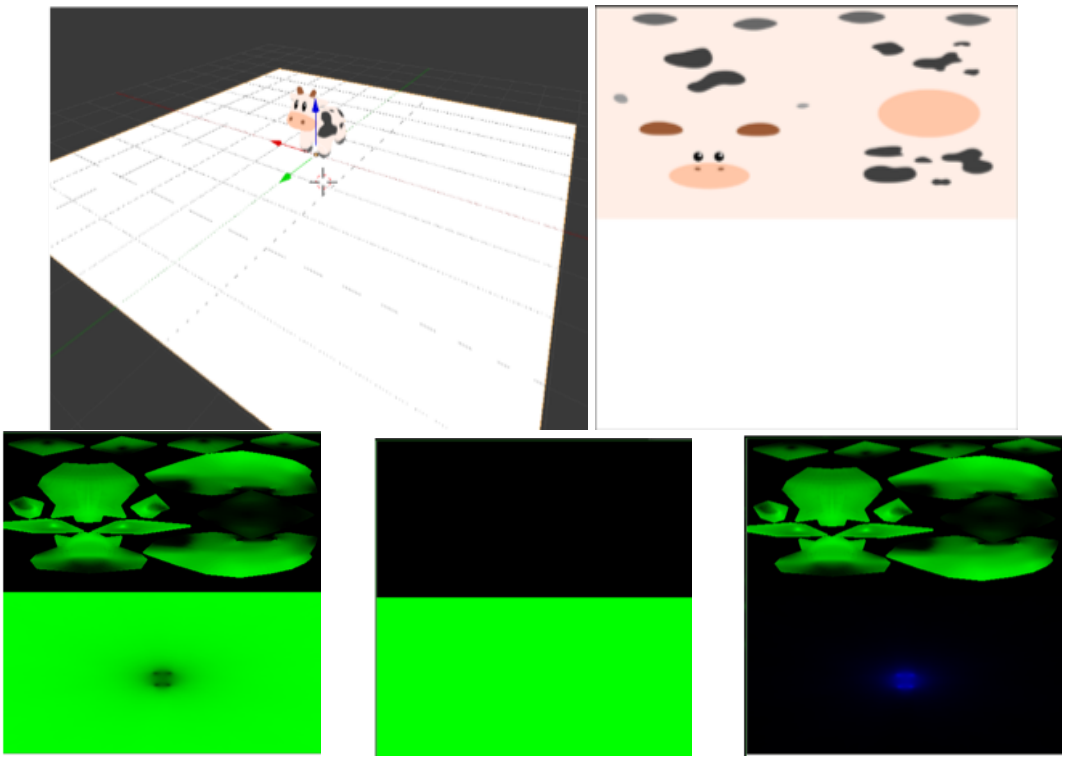


Figure 4.9: Top left: Mesh, consisting of virtual object on plane representing real surface below it. Top right: albedo texture, with virtual object texture mapped to top half of texture space, and real world plane to bottom half. Bottom: PRT textures used to calculate differential PRT. SH band 0 shown, with green representing positive values, blue negative and black zero.

data.

Figure 4.11 shows an example environment map rendered using this information. The cubemap is visualised as a net.

Figure 4.12 shows the final augmented image, containing a reflective virtual sphere.

It can be seen that often, the lower walls and floor of a room will not be visible to the fisheye camera during typical use. However, the surface under the inserted virtual objects, such as a table or floor tends to be captured in the dense reconstruction, meaning that a complete environment map can be produced.

4.4.2 Comparison to Physical Light Probe

In order to test the effectiveness of the approach, a qualitative comparison was performed between virtual and real reflective spheres. The shadow cast by the virtual

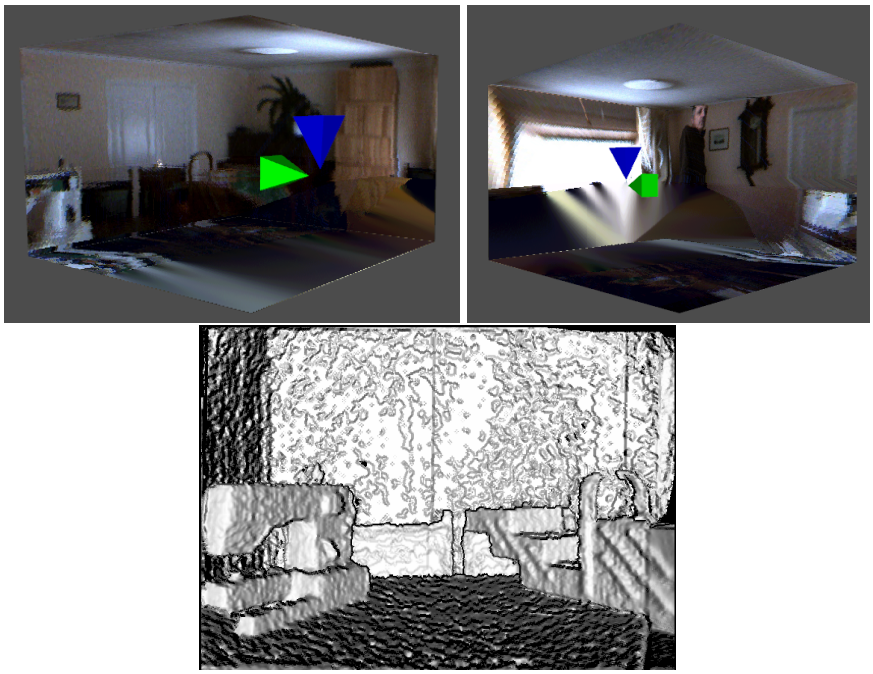


Figure 4.10: Coarse and dense models captured by the application, in the room of a home. Top: Two views of the textured coarse model. Locations of the RGBD and fisheye cameras are shown as blue and green frustra, resepectively. Bottom: dense reconstruction (raycasted).

Figure 4.11: Environment cubemap produced by proposed approach. Rendered using the coarse and dense models shown in figure 4.10.

Figure 4.12: Image augmented with virtual sphere, rendered using the environment map from figure 4.11.

sphere was rendered using differential PRT, as described in section 4.3.8. In the examples shown, the capture process was performed, and a virtual sphere rendered. Afterwards, a real sphere of the same dimensions was placed in the same location, and another image taken. The two images were taken using the same camera pose, by attaching the camera to a sturdy tripod. The results are shown in figure 4.13.

The two spheres appear quite similar - both the nearby objects and the more distant components of the scene, such as the ceiling lights, are reflected in the correct locations on the virtual sphere. Like the real sphere, the virtual sphere also casts a shadow on the table, and this shadow is reflected on the sphere. There are some differences, particularly in the shadowed region under the sphere. In the virtual example, the shadow is softer. This is partly a consequence of the use of spherical harmonics (which can only represent low-frequency illumination) and partly a consequence of the low dynamic range of the camera pair. The reflections on the lower edge of the virtual sphere are also slightly incorrect, causing the thin bright band on the lower edge of the virtual sphere. This is a consequence of using an environment

Figure 4.13: Comparison between a virtual reflective sphere and a real chrome sphere. Top: virtual sphere. Bottom: real sphere.

cubemap, which was rendered from the centre of the sphere.

Additionally, the dense model has not yet fully captured the nearby real scene, so some parts of the table are missing in the reflection. Finally, the reflections on the real sphere are slightly blurred, due to the limited dynamic range and focal depth of the camera, as well as the imperfect surface of the real sphere.

Figure 4.14 shows a challenging situation for this system. In the middle image, taken as the application started, the purple side of the box, to the right of the sphere is not reflected. This is because the RGBD camera did not obtain depth values for this side of the box, and it was not added to the dense reconstruction. The lower

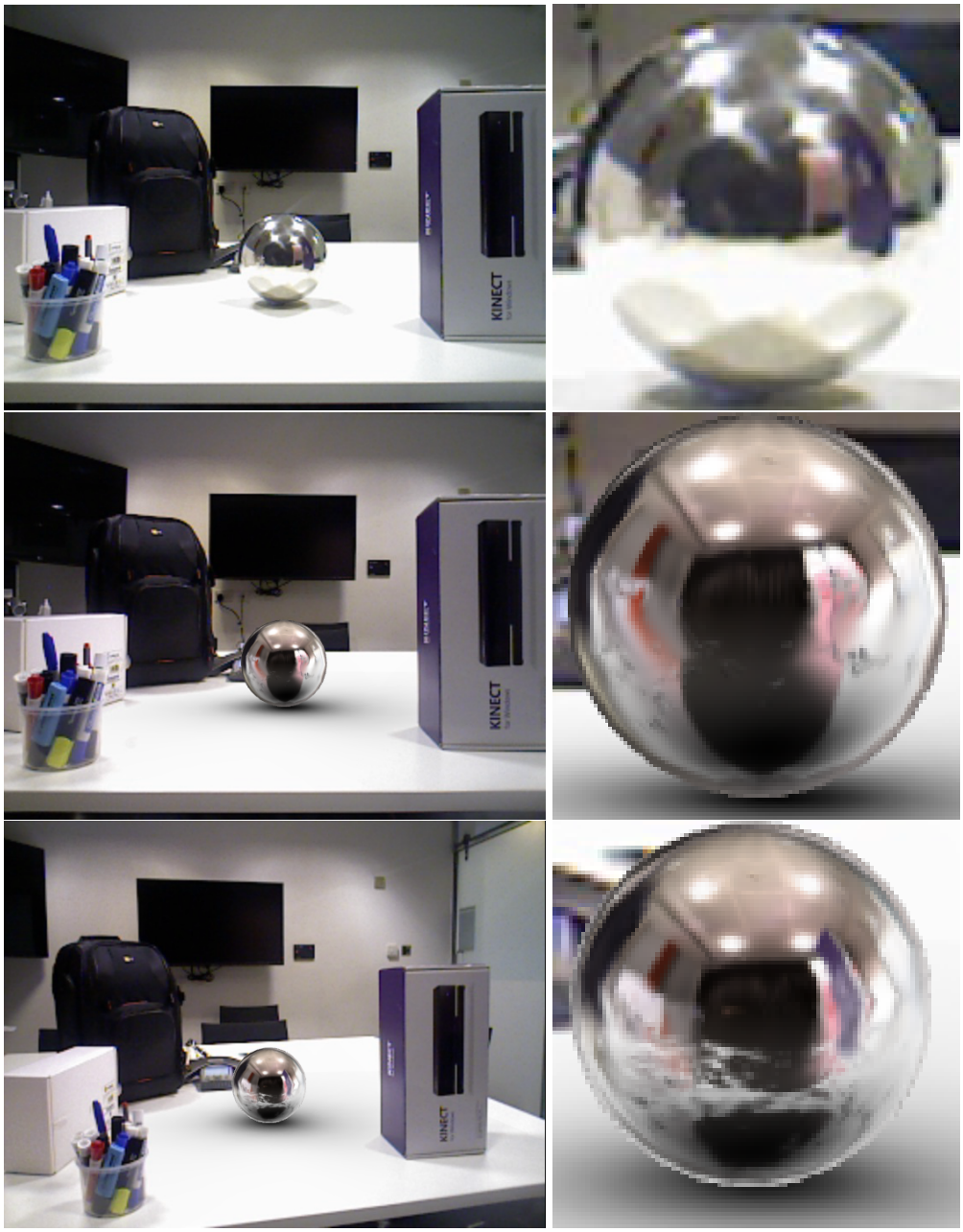


Figure 4.14: Example scenario where surfaces are not initially visible to the RGBD camera. Top: Image of scene with real reflective sphere, for comparison. Middle: AR image with virtual sphere, on first frame of sequence. Bottom: AR image after moving camera, exposing more of the real scene to the RGBD camera.

image shows an example after the camera has been moved to the left: the side of the box facing the sphere is now visible, and is reflected properly. As the dense model becomes more complete, the environment maps generated become gradually more accurate, as can be seen in the bottom image.

4.4.3 Baseline Approach

The approach was also compared to a simpler baseline approach, without the coarse model approach proposed here. This approach renders the fisheye image into the virtual object's environment map each frame, after applying a rotation based on the current fisheye camera pose. The environment map is not cleared after each frame, allowing it to build up as more of the room is observed. This approach implicitly assumes that the environment observed by the fisheye camera is distant from the object, and does not account for the translation between the virtual object and the camera pair. Example frames from this comparison are shown in figure 4.15.

These frames were taken from the end of a short sequence during which the camera was moved around the virtual content. Even so, one can see that the environment map generated by the baseline approach is still very incomplete, due to the lack of inpainting and dense modelling. The lower half of the cubemap has yet to be observed, the geometry is incorrect (for example, the window is too large) and the reflections of the sewing machine, chest and table are not present. An example is also shown with the dense reconstruction added. This is an improvement, but there are still incomplete regions, and the reflections of distant objects are still geometrically inaccurate.

4.4.4 Timing Analysis

In order to assess the performance characteristics of the approach, the application was executed, and the tasks involved in rendering a single frame were individually timed. The input sequence was taken from the rooms shown in figure 4.5. The virtual object rendered was the teapot from figure 4.1.

The timings shown were obtained on the CPU, however, many components of the system execute on the GPU. In order to obtain meaningful timings, the GPU

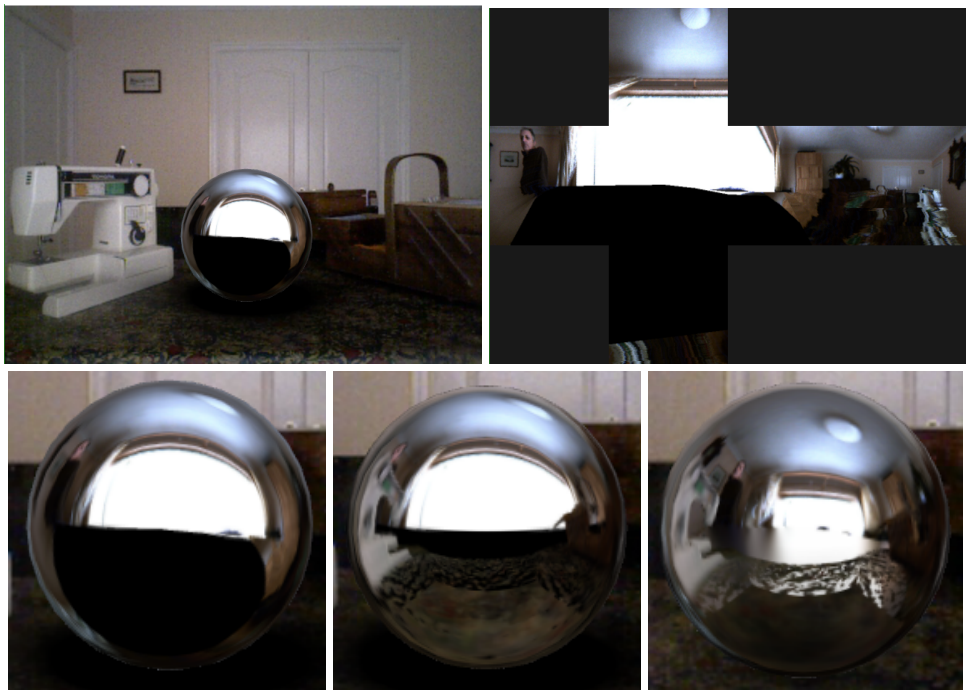


Figure 4.15: Comparison of the proposed approach with a simpler, baseline approach. The sequence used is the same as that used in figures 4.10, 4.11 and 4.12. Top left: AR Image, using baseline approach. Top right: Environment map, using baseline approach. Bottom left: Closeup of virtual sphere, using baseline approach. Bottom middle: Closeup of virtual sphere, using baseline approach with dense reconstruction. Bottom right: Closeup of virtual sphere, using our approach.

was explicitly synchronised after each of these stages. These timings were taken on a desktop PC, using an Intel i7-4970 CPU and an Nvidia GTX 1080 GPU.

Table 4.1 contains the results. Note that, for simplicity, the less time-consuming stages were combined into the "Other" category. The main component of this is swapping the framebuffers (including vsync). This frame completed in 47.42ms, corresponding to a framerate of 21.1fps.

The brightness estimation stage is the most time-consuming stage. The current implementation of this involves rendering a difference image, and summing it serially on the CPU. The performance could be improved significantly by parallelising or moving to the GPU. It could be further optimised by sparsely sampling, rather than using all pixels in the coarse model texture. The undistortion and SH projection stages are also currently implemented serially on the CPU, and could benefit from a parallel GPU implementation.

Task	Time (ms)	Percentage
SLAM Update	3.16	6.7
Undistort fisheye image	8.55	18.0
Upload images to GPU	0.84	1.8
Render occlusion mask	0.61	1.3
Estimate brightness	21.37	45.1
Update coarse model	0.29	0.6
Render cubemap (coarse model)	0.91	1.9
Render cubemap (dense model)	7.46	15.7
Project cubemap to SH	0.38	0.8
Render scene	0.228	0.5
Other	3.61	7.6
Total	47.42	100.0

Table 4.1: Timing breakdown of a typical frame rendered by the application.

Rendering the dense model component of the cubemap is also quite time-consuming. Whilst rendering the coarse model just involves rendering a textured mesh, adding the dense reconstruction requires a costly colour raycast of the InfiniTAM volume. This performance of this stage also depended heavily on the cubemap resolution; in this example, with 256x256 cubemap faces, it was relatively fast. At higher resolutions, it consumed over 50% of the total time.

4.5 Conclusion

A system was developed to synthesise environment maps for virtual objects in indoor mixed reality applications. The system used a single, self-contained device containing two cameras. The environment maps were used to render reflective virtual objects, and the results were compared qualitatively to real mirror surfaces. The system was shown to be able to produce detailed environment maps, which could be used to render virtual objects with high-frequency lighting effects such as reflection and refraction.

Since the coarse model updating and environment map rendering take place in real time, the presented approach can handle a number of dynamic changes. Changes in the surrounding environment such as a video playing on a nearby television screen, a change in the weather outside or a person walking past can be handled correctly. The virtual objects are also capable of changing their position arbitrar-

ily within the real scene. This opens up new possibilities for more dynamic and engaging mixed reality content.

In this system, the reconstructions generated were used to generate environment maps. These are efficient to render, but are not strictly geometrically accurate, unless the surrounding real scene is sufficiently distant from the virtual object. It would be possible to use a more advanced approach, such as reflection mapping with parallax [156] or multi-perspective rendering [57], to provide more realistic results. Alternatively, the coarse and dense models could be used as input to other rendering techniques such as ray-tracing.

Environment maps also perform poorly when rendering flat, planar surfaces. When such a virtual object is to be rendered, it would be preferable to render the reflection using a camera placed at the reflected (virtual) viewpoint [30]. This could be added to enable the system to render a wider variety of objects, such as virtual mirrors.

There are also ways in which the system presented here could be enhanced, to improve ease of use. For example, by generating the coarse model automatically, or by using the fisheye camera or a small inertial measurement unit to improve camera tracking.

The dense model only contains surfaces observed by the RGBD camera. If the user does not capture enough of the region around the virtual objects, incomplete parts of the dense model may be visible in the environment maps (see fig. 4.14). A completion method such as [39] could be added to address this problem.

The approach presented in this chapter focused on using camera tracking and 3D reconstrution to solve the geometric issues involved in generating environment maps using our hardware setup. However, photometric accuracy was not addressed in detail. The following chapter explores extending this earlier approach to capture the full dynamic range of the environment. The aim of this is to reduce artefacts such as the excessively soft shadow in figure 4.13, and allow for more accurate rendering of diffuse virtual objects.

Chapter 5

High Dynamic Range Illumination Capture

5.1 Introduction

As briefly discussed in the conclusion of chapter 4, the quality of the results produced using the system was limited due to only capturing low dynamic range (LDR) information about the real scene. Only capturing information within a limited dynamic range means that certain details of the real scene cannot be captured. For example, real light sources are frequently so bright that they cause pixels in a typical LDR image to be saturated, meaning it is not possible to infer the true intensity of the light source. Conversely, other parts of the scene may be dark enough to be below the black point of the image sensor, again meaning that detail cannot be recovered. The saturation of pixels corresponding to light sources led to problems such as the unrealistic reflections of real light sources visible in the virtual sphere rendered in figure 4.13.

In order to improve upon the previous approach, it was adapted to instead capture richer high dynamic range (HDR) information about the real scene. This would help to correct these issues, in addition to opening up the possibility of using more accurate HDR rendering techniques to exploit the real scene information. Furthermore, the previous approach required the two cameras to have fixed auto exposure and white balance, wheras the newer approach allows them to adapt to better fit the

lighting conditions in the real scene.

5.2 Approach Overview

The overall approach was structured in a similar way to the earlier approach of chapter 4, but a significant number of modifications were made to convert the system to both capture HDR colour values and use these values in the rendering process.

Figure 5.1 gives an overview of the approach, indicating which components use HDR colour values, and which use LDR values.

5.3 Approach Detail

The individual components of the approach will now be described in more detail, with particular focus on changes made from the earlier approach of chapter 4.

5.3.1 Hardware Setup

The improved HDR version of the system used nearly the same hardware setup as the earlier system described in chapter 4 (see figure 4.3). The main change was to replace the earlier camera, a Point Grey Chameleon CM3-U3-13Y3C-CS with the similar CM3-U3-31S4C-CS model. This new camera was identical in size and shape, but had a slightly higher image resolution and an exposure bracketing feature, which was used for the improved approach. The same Lensagon fisheye lens was fitted to this new camera, which was again attached to the Asus RGBD camera. Since the camera had been changed, the calibration process detailed in chapter 4 was repeated after fitting the new camera.

Both of the cameras in the system produced typical 8-bit LDR output. In order to capture HDR colour information, the bracketing technique was used [29], making use of the camera's bracketing feature. This feature enabled it to cycle through a series of 4 different manually selected shutter and gain settings, whilst capturing frames at normal video framerate.

In contrast to the previous approach, the fisheye camera was also set up to produce linear output, disabling all features such as gamma correction and white balance. This created some issues with capturing footage in indoor environments,

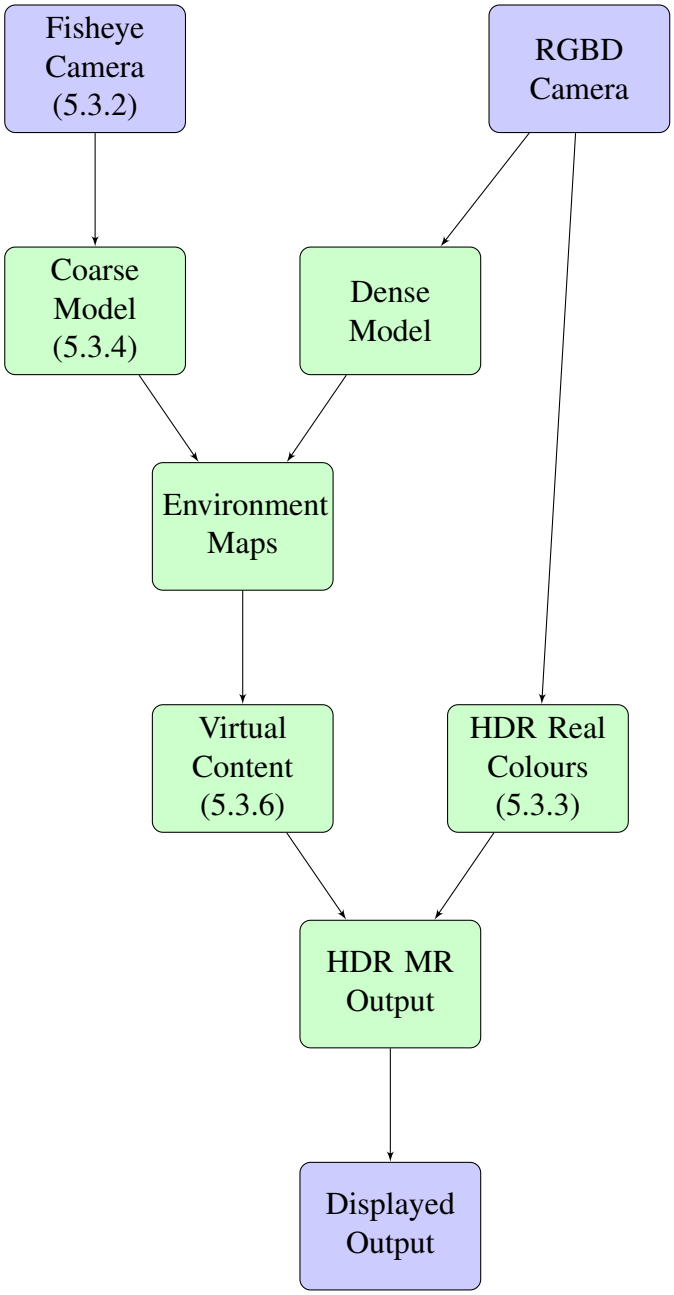


Figure 5.1: System overview flowchart, with LDR content labelled in blue, and HDR content in green.

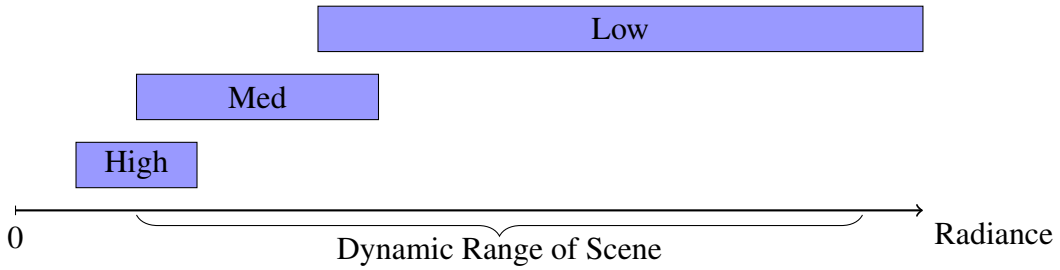


Figure 5.2: Illustration of a suitable set of 3 exposure settings, capable of capturing the full dynamic range of a real scene.

due to mains freqency flickering of light sources. At 50 or 60 Hz, this flickering is too fast to be visible to humans, but if captured with a camera at a different framerate (e.g. 40 Hz), it can create lower frequency alisasing which is very visible. To mitigate this problem, the framerate of the camera was fixed at mains frequency (50Hz in the UK). This did not completely solve the problem, as the flicker would still cause us to estimate slightly inaccurate radiance values, but it greatly reduced these inaccuracies.

5.3.2 Selecting Fisheye Exposures

As mentioned above, the fisheye camera's bracketing mode cycled through 4 exposure settings. A method needed to be developed to select these 4 settings to capture the full intensity range of the real environments. For this reason, a simple adapted auto-exposure approach was implemented.

When designing the approach, a number of sample images were taken in a few typical target environments, in which the system might be used (i.e. indoor environments). After measuring the radiance difference between the darkest and brightest parts of these images, it was estimated that 3 exposures would be sufficient to capture the brightest (e.g. windows on sunny days) and darkest (e.g. areas under tables) parts of the scenes, whilst still leaving an overlap between the exposures. This overlap would ensure that there would not be areas which fall between two exposure ranges, meaning their radiance cannot be determined. Figure 5.2 shows an illustration of how 3 suitably selected exposure settings could capture the dynamic range present in a real scene.

For this reason, the fisheye camera's four exposure settings were set in a low,

medium, high, medium order (LMHM). This meant that every 4 frame cycle would capture the full exposure range of the scene. The extra medium exposure at the end of the cycle was selected as typically the medium exposure captures the largest number of valid pixels (i.e. not saturated or zero).

A simple algorithm was developed to dynamically select these three exposure settings. Since the camera was configured to produce linear output, the only two factors which could be modified were exposure time and gain. Furthermore, fixing the framerate at mains frequency $f = 50Hz$ meant that the exposure time was bounded above by $\frac{1}{f} = 0.02s$[1]. In the following, we define exposure as $E := gt$, where g is gain and t is exposure time. Our camera gave gain in decibels g_d, so this was converted to a linear scaling factor $g = 10^{\frac{g_d}{20}}$.

The medium exposure was set to a fixed value, which we found to be reasonable for the scenes where the approach was used. The high and low exposure settings were chosen based upon the number of saturated and zero pixels in the most recently captured images. If the number of saturated pixels in the low exposure image exceeded a threshold, the exposure was lowered by one stop. If there were no saturated pixels at all, the exposure was increased by one stop. Similarly, if the number of zero pixels in the high exposure image exceeded a threshold, the exposure was increased, and if there were no zero pixels it was decreased (again a stop at a time). The high and low exposures were always at least one stop higher or lower than the medium exposure, respectively.

The number of saturated and black pixels were counted on the GPU, using an efficient reduction which meant that this step did not consume a significant proportion of the computation time. In future, if this step were to be optimised, it would probably be sufficient to just sample a subset of the pixels in the image and estimate the proportion of saturated or black pixels, but this was not explored here.

These criteria were used to choose appropriate exposure values E. This still leaves the issue of selecting appropriate gain and exposure time settings g, t so

[1]Mains frequency varies between different countries (and between eastern and western Japan). If this approach were to be used in an area where mains frequency was 60Hz, it would be necessary to fix the framerate to $60Hz$ and limit the exposure time to $\frac{1}{60}s$.

that $E = gt$. Here, g and t were selected to maximise exposure time t, subject to the constraint that $t < \frac{1}{f} = 0.02s$. This helped to avoid the issue of high gain amplifying sensor noise. In principle, setting t too high could result in blur when the camera or scene objects move. This was not an issue in this case as the very short focal length of the fisheye lens meant there was no noticeable blur at the maximum exposure time of 0.02s.

In summary, the algorithm worked as follows:

1. If there are too many saturated pixels in the L exposure, decrease the exposure.
2. If there are no saturated pixels in the L exposure, increase the exposure.
3. If there are too many zero-valued pixels in the H exposure, increase the exposure.
4. If there are no zero-valued pixels in the H exposure, decrease the exposure.
5. Maintain a separation of at least one stop between the exposure settings.
6. Always select the highest exposure time and lowest gain possible for a given exposure value.

5.3.3 Convering Input Images to HDR

Thanks to the linear output of the camera, we are able to convert the input fisheye pixel values to HDR values proportional to radiance by simply dividing by the exposure E, defined in the previous section. This does make the assumption that there is no variation in response to incoming radiance over the fisheye image. This assumption might be incorrect for a variety of reasons, including variation between the pixels in the sensor and lens artefacts such as vignetting. Since we used a wide-angle lens and set the aperture to be narrow, we were able to remove much of the vignetting by cropping a small part of the boundary of the image circle.

As noted above, the HDR images only contain values proportional to the true radiance; that is, $v_i = KL_i$, where L_i is the true radiance at the pixel and K is an

unknown constant. Note that whilst K is unknown, it is constant across each image and between all captured images. The true value of K depends on a number of factors including the focal length, aperture and transmissivity of the lens and the response of the sensor. For this application it is sufficient to know v_i as the final output will be converted back to LDR for display in any case.

Converting the RGBD images to HDR was more challenging. In contrast to the previous approach, the RGBD camera was set up with both auto exposure and auto white balance enabled. This enabled the main images used to create the augmented output to adapt to the lighting conditions and remain aesthetically pleasing and easy to see. Since the RGBD camera used here did not supply exposure information when these features were enabled, the exposure was inferred indirectly by comparing pixels in the region of overlap between the two cameras. This was then used to store corrected colours in the surfel model, and to correctly add augmented content to the image.

The model we fit is the same as that used by Zhang et al. [158] and Rohmer et al. [117]. This consists of inverse gamma correction using a constant $\gamma = 2.2$, followed by linearly multiplying each channel of the image by a separate constant $e := (e_r, e_g, e_b)$, jointly modelling exposure and white balance. Given a set of pixel values from matching locations in the inverse gamma corrected RGBD image l_i and HDR converted fisheye image f_i, the exposure/white balance factor e is estimated as $e \approx \frac{\sum f_i}{\sum l_i}$.

When finding the matching pixels l_i and f_i, only the medium fisheye exposure is used, as the other exposures typically contain many saturated or zero-valued pixels. Matches are found using the following procedure at starting pixel in the RGBD image.

- If a valid depth value is available here and the colour is not saturated or zero-valued, reverse-project to the corresponding 3D point.

- Project this 3D point into the fisheye image. If it projects to a valid image location, sample the fisheye image.

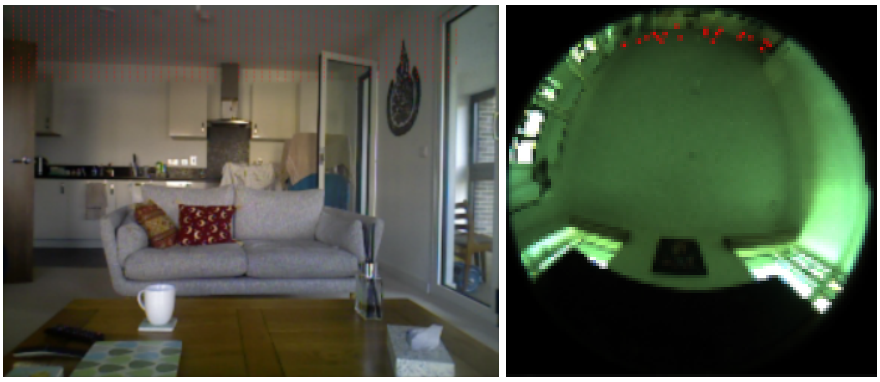


Figure 5.3: Sample locations used to estimate AE/WB of RGBD image, visualised as red dots. Left: RGBD colour image. Right: Fisheye image (medium exposure). Here, the red dots correspond to the locations of the successfully reprojected samples only - those that correspond to locations with invalid depths (for example) are not shown.

- If the fisheye sample is also not saturated or zero-valued, add the fisheye and RGBD colour samples to a list.

In practice, starting from a small proportion of locations spaced out over the top of the RGBD image is sufficient to obtain a reasonable estimate of e in most cases. Figure 5.3 shows a visualisation of the sampling pattern used in the final implementation. In situations where very few matches were successfully found, the new estimate for e is unlikely to be accurate and the one from the previous frame is used instead. Since the fisheye and RGBD images were captured at different times, an estimated pose for the fisheye camera is interpolated using the current and previous RGBD poses. The process of finding and summing the matches is implemented mostly in a compute shader on the GPU, providing some improvement in performance.

5.3.4 Updating the Coarse Model

The coarse model updating process in this approach is similar to that described in chapter 4, but some adaptations had to be made when moving from LDR to HDR values. A naive approach to updating the coarse model might involve simply updating all texels with the latest HDR-converted fisheye image every frame. However, this is likely to give inaccurate and noisy results. The inaccuracy stems from using saturated or zero-valued pixels, and the noise results from using values from the low

exposure image when this is not appropriate.

Instead, two different approaches were implemented to update the coarse model. These balanced the need for timely updating with accuracy and avoiding noise. The approaches each had different strengths and weaknesses which are explored in more detail later in the chapter.

The first approach (referred to here as the "best score" approach) updated observed texels using the "best" recorded colour during each 4-frame cycle. This approach was similar to the techinque proposed by [29] for still images. Here, "best" is defined using a simple score function, which favours colour values from the middle of the exposure range of each image. This function is defined as the hat function:

$$S(x) = \begin{cases} x & x \in [0, 0.5) \\ 1 - x & x \in [0.5, 1] \end{cases} \tag{5.1}$$

A double-buffering-style approach using two separate coarse model textures was used to avoid showing partially updated coarse model textures to the user. The best score so far for each pixel in the current cycle was stored in another texture of the same size, and the updating process was carried out on the GPU using a compute shader.

The best score approach produced good quality results for fairly static scenes, but resulted in the coarse model updating at a lower framerate and caused occasional ghosting artefacts when the real environment changed during a 4-frame cycle.

The second approach (the "Kalman filter" approach) updated all observed texels every frame, but carried out a weighted update, using a simplified Kalman filter approach. This had the effect of carrying out a smoothed update, where the update considered the expected noise in the new supplied value.

The Kalman filter used here was applied to each channel of each pixel in the image. Two colour textures were used; one to store the current colour estimate, and the other to store the corresponding error covariance. The update rules for the

implemented Kalman filter are given below. In the following, G is the Kalman gain, C is the current covariance, S and P are the sample and process covariance, V is the estimated radiance value and I is the image sample (converted to HDR). The subscripts n and $n+1$ indicate the fisheye frame number. All values are 3-vectors.

$$G_{n+1} = \frac{C_n + P}{C_n + S_{n+1}} \tag{5.2}$$

$$V_{n+1} = V_n + G_{n+1}(I_{n+1} - V_n) \tag{5.3}$$

$$C_{n+1} = (\mathbf{1} - G_{n+1})C_n \tag{5.4}$$

The process noise P was set to a constant value. The sample noise S was estimated using a simple sensor noise model. In practice, the process noise P was chosen manually, and served as a way to control the behaviour of the updating process, with smaller values of P resulting in a smoother output at the cost of slower response to changes in lighting.

The Kalman filter approach produced good results in many cases, but had the effect of smoothing out dramatic lighting changes. For example, when a real light was switched off, in the coarse model it would instead gradually dim over a period of about half a second. Although increasing P could address this problem to an extent, if P was set too high this would result in flickering, noisy output.

5.3.5 Updating the Dense Model

The dense model was reconstructed and updated in a very similar way to that used in the previous approach, making use of the InfiniTAM dense RGBD SLAM system. In this instance, a newer version of the library was used (InfiniTAM v3), as this had now become available.

The InfiniTAM library was originally capable of capturing reconstructions with low dynamic range colour values, stored as 8-bit unsigned values. For this reason, we modified the library to instead store colours as 32-bit floating point values. This necessitated changes to the model representation, updating, rendering and tracking components of the system.

It should be noted that whilst InfiniTAM v3 is capable of both volumetric and surfel-based reconstructions, the examples shown here all use the volumetric representation. This was chosen to allow for easier comparison with the previous (LDR) approach, however the presented HDR illumination method is capable of using either representation.

5.3.6 Rendering the Virtual Content

Whilst the overall rendering approach was similar to that described in chapter 4, adaptations were made to produce HDR output. The overall rendering procedure began by first rendering an environment map from the virtual object location. This environment map was then projected into SH lighting coefficients, and the lighting coefficients and environment map were used to render the added virtual content. This rendered content and the input RGBD colour image were used to produce the final MR output.

The environment map was rendered in a similar way to chapter 4, by first rendering the coarse model and then rendering the dense reconstruction on top. In this approach, however, the environment map stored HDR values in a 32-bit floating-point cubemap.

The virtual content was then rendered into a HDR texture, which again stored 32-bit floating point colours, which could be positive or negative in value. In addition to rendering the virtual objects themselves, differential PRT was also used to render their influence on the real environment (e.g. shadows cast onto real surfaces under the virtual object). Each pixel in the HDR texture was labelled as being in one of three categories: real, virtual or sum. These indicated how the pixel should be composited into the final MR image.

Pixels in the "virtual" category were generally located on virtual objects, and would completely replace the captured image at that pixel. At pixels in the "real" category, the value from the captured real image was used. At pixels in the "sum" category, the real and virtual pixels were added together to produce the final result. This category was assigned to pixels where a virtual object affected the real environment, for example where it shadowed or produced a caustic on a real surface.

Note that since the virtual object texture could store positive or negative values, it was possible to increase or decrease the real observed value.

A compute shader was used to composite the virtual content HDR texture and HDR-converted real RGBD image. This produced an MR image in the HDR colour space. Before displaying the image to the user, it was converted back to LDR, by applying the auto exposure, white balance and gamma of the RGBD colour camera. In this manner, the image appeared to have been observed by the real camera.

5.4 Results

Figure 5.4 shows examples of the technique being used to render a variety of virtual objects in two different real scenes.

Perhaps the main advantage of this HDR method over the earlier LDR approach of the previous chapter in practice is removing the need for manual adjustment. In order to capture the examples shown in chapter 4, it was necessary to tweak the white balance and auto exposure of the fisheye camera to produce similar output to the RGBD camera. These settings then had to be locked on both cameras, in order to ensure that they did not deviate from one another. Finally, a manually chosen scaling factor had to be applied to the SH lighting in order to produce realistic-looking results. With the new HDR approach, none of this is necessary. As an added advantage, the exposure and white balance of the main camera no longer have to be fixed, meaning they can adjust to produce more pleasing output images.

The shadows produced by the new approach were generally slightly improved over the LDR approach of chapter 4, but still often appeared too soft. This was likely a consequence of using SH lighting - after projection to the relatively small number of SH coefficients used here, the light sources were somewhat blurred, resulting in softer lighting. In order to take full advantage of the new HDR lighting model captured by this approach, it might be necessary to use a different rendering technique capable of producing harder shadows. Ray-tracing could provide one option, but reasonable results might also be obtained by fitting a number of virtual point lights to the coarse model and using shadow mapping.

Figure 5.4: Examples of MR frames rendered using the HDR approach, containing a variety of different virtual objects and real scenes. Top left: virtual wooden artist's mannequin, with diffuse material. Top right: virtual teapot, with mirror reflective surface. Bottom left: virtual bunny, wtih diffuse material. Bottom right: virtual dragon, with slightly rough blue metallic material.

5.4.1 Comparison with Physical Light Probe

The system was compared against a real physical light probe (a chrome sphere) using the same approach as outlined in section 4.4.2. The results are shown in figure 5.5.

There are some slight differences between the two images that do not result from the approach itself - in particular, the framing of the image is slightly different, and the exposure of the camera has also slightly changed. Although the author attempted to capture images which were as similar as possible by making use of a quick-release tripod, the position of the tripod has changed slightly between the two images.

There are some geometric differences between the reflections in the two images - the reflection of the table does not extend quite as far in the right hand image, which is due to this part of the table having not yet been fully reconstructed. Objects

Figure 5.5: Comparison of a virtual reflective sphere rendered using the HDR approach detailed in this chapter with a real chrome sphere of the same size. Above: Real sphere. Below: Output of approach detailed in this chapter.

very near the sphere, such as the pens on the table and monitor in the background, have slightly inaccurate reflections. Similarly to the earlier approach of chapter 4, this is the result of the use of environment maps to render the reflections, and these inaccuracies could potentially be removed by using ray-tracing against the dense model instead.

The size and position of the ceiling light reflections is very similar to the real sphere. There is a slight difference in the brightness of the ceiling in the reflection, which could be a consequence of the estimated AE and WB settings of the forward facing camera being slightly incorrect.

The reflection of the wall behind the camera in the centre of the virtual sphere is different to that in the real sphere, as a consequence of this area being inpainted. After more time moving the real camera around and observing more of the real environment, this area would be filled with real observations, resulting in more accurate reflections.

5.4.2 Coarse Model Updating Approaches

The two methods for updating the coarse model outlined in section 5.3.4 were compared to one another. The examples shown here were produced by recording footage first, and then supplying this to the application twice - once using the best-score updating approach, and once using the Kalman filter-based approach.

Both methods produced good results where the real scene was static, and the tracking of the camera pair was accurate. In situations where the real scene changed, or incorrect tracking caused data to be integrated into incorrect parts of the coarse model, however, both methods exhibited artefacts. The nature of these artefacts was different with each approach.

The best-score approach only updated the coarse model once in each four-frame cycle. Since our upward-facing fisheye camera had a framerate of 50fps, this resulted in an effective update rate of 12.5Hz for the coarse model texture. This was significantly lower than the 30Hz framerate of the RGBD camera used to display the AR input, and this discrepancy was noticeable in certain situations. For example, if a person walking through the room were visible in the reflection from a virtual object, their motion would appear "jerky" due to the low update rate.

The best-score approach also implicitly assumes that there is no significant change to the real environment during each four-frame LMHM bracket. That is, it is assumed that each of the four images samples the same real lighting environment (albeit with different exposure and gain settings). This is not the case where there is motion in the real environment, however. Figure 5.6 shows an example of the output which can result from this problem. Here, a person stands in front of a window, holding both arms out in a T-pose and moving them up and down. The arms are moving rapidly, so their position is different in each of the bracketed frames. Consequently, the arms are not properly represented in the coarse model texture, and there are visible artefacts at the different locations of the arms in the input images.

The Kalman filter-based approach suffered from a different kind of artefact in the presence of rapid motion. Using this approach, the coarse model was updated

Figure 5.6: Example of the output of the best-score approach where rapid movement is present in the real scene. Top: bracketed fisheye frames used to update coarse model (low, medium and high exposures). Bottom: close-up of person in coarse model.

every fisheye frame (i.e. at 50Hz), meaning that this approach did not suffer from "jerky" motion. However, since each pixel is more gradually replaced with new observations, rapid motion tends to lead to artefacts similar to motion blur in the output. Figure 5.7 shows an example using the same input footage as figure 5.6, but now using the Kalman updating method. Another consequence of the Kalman filter-based approach is that the model updates more slowly where there is a large change in radiance. This causes more motion blur in this example, where a person wearing dark clothing is moving in front of a bright window. Furthermore, since the filter is applied independently to each colour channel, the motion blur can exhibit coloured artefacts, such as the yellowish motion blur under the person's left arm in figure 5.7.

An advantage of the Kalman approach is that the parameters of the filter can be adjusted to tune the balance between updating more rapidly (and hence avoiding motion blur artefacts) and updating more slowly, which helps to minimise temporal

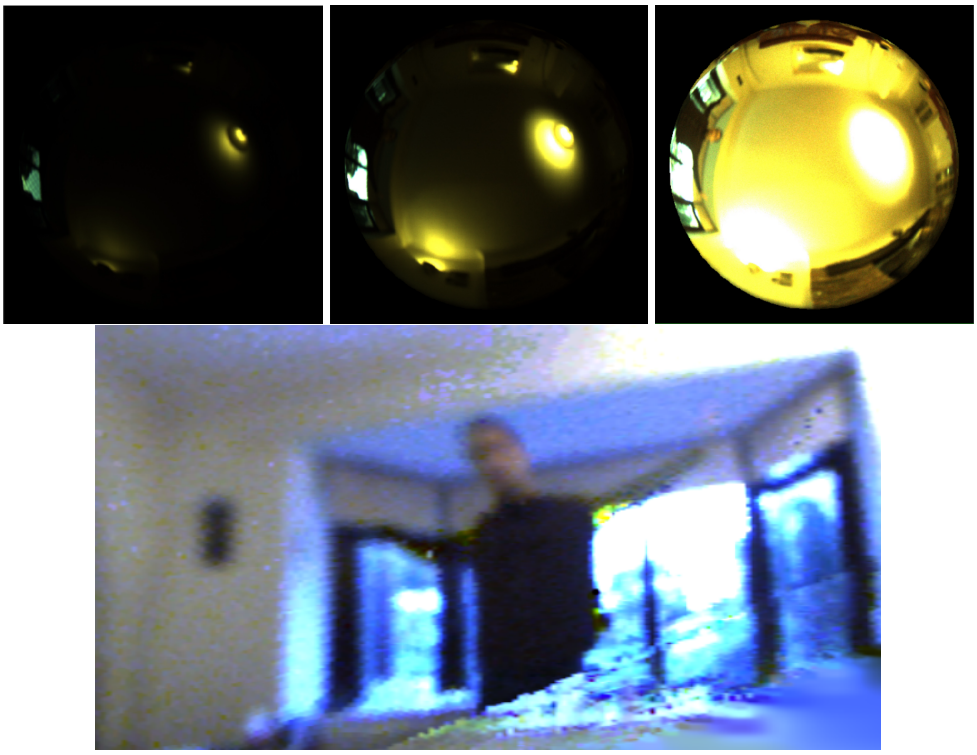


Figure 5.7: Example of the output of the Kalman approach where rapid movement is present in the real scene. Top: bracketed fisheye frames used to update coarse model (low, medium and high exposures). Bottom: close-up of person in coarse model.

noise in the output. The optimal choice of parameters might be different in different scenarios, depending upon how rapidly the real environment might be expected to change. In this example, captured at dusk, the real environment was fairly dim, resulting in large amounts of noise in the input fisheye images. The Kalman filtering approach helped to remove some of this noise from the output - this is most visible on the wall to the left of the window.

To summarise, the best-score approach produced reasonable results but suffered from unnatural-looking artefacts and a "jerky" response to rapid motion in the real scene. The Kalman approach did not suffer from "jerky" motion, but instead tended to exhibit artefacts appearing similar to motion blur. It was also capable of removing some of the temporal noise present in images captured in low-light conditions.

On the whole, the authors considered the Kalman approach more successful, and this is the method used in the other examples shown in this section. The choice

between the two approaches is somewhat subjective, however, and there may be situations in which the best-score approach might be considered to provide more pleasing results.

5.4.3 Comparison with LDR Approach

The revised, HDR approach was compared to the earlier LDR approach of chapter 4. In the example shown, separate LDR and HDR sequences were recorded. This was necessary, as the LDR and HDR approaches required different inputs. Specifically, the LDR approach required LDR video from the RGBD and fisheye cameras, and the HDR approach required bracketed HDR video from the fisheye camera, along with exposure information for each frame.

In order to provide a fair comparison, in each of the examples both sequences were filmed one after the other in the same location, without changing the geometry or lighting conditions of the real environment. At the end of each sequence, the camera pair was placed on a tripod, which was fixed in place. This ensured that the comparison frames were captured from the same camera location.

It is important to note that when recording the examples for the LDR approach, extra care needed to be taken to manually adjust the white balance and exposure of the RGBD and fisheye cameras to be similar, and to manually tweak the shadow intensity for the SH lighting.

Using the new HDR approach has given some small improvements to the output - most noticeably, the reflections of the ceiling lights are more accurate - in the LDR example, differences between the response of the fisheye and RGBD cameras have caused the lights to appear too dim in the reflection. In the HDR approach they appear brighter and larger, and are more similar to the reflections in the real sphere (see figure 5.5.

It was also hoped that moving to HDR, and capturing the true brightness of the light sources might result in sharper, more realistic shadows. Unfortunately, this does not seem to be the case here - in the HDR example, the shadow is still too soft. As mentioned in section 5.4, this is likely a consequence of the rendering method used (PRT, using 5 bands of SH) being unable to make full use of the information

Figure 5.8: Comparison of a virtual reflective sphere rendered using the LDR and HDR approaches detailed in chapters 4 and 5. Above: Virtual sphere, rendered using LDR approach of chapter 4. Below: Virtual sphere, rendered using HDR approach of chapter 5.

available. In future work, it would be interesting to explore using the coarse and dense models as input to other rendering approaches.

Again, it should be noted that obtaining the shadow shown here using the LDR approach involved some manual adjustment, wheras the HDR approach was able to produce a plausible shadow without any intervention.

5.5 Conclusion

In this chapter, an improved version of the MR illumination capture system of chapter 4 was introduced. This system added the ability to capture and make use of HDR lighting information from the real environment.

Successfully capturing and using HDR information required a number of significant changes to the earlier approach. These included changes to the coarse model and the method for updating it, in addition to changes to the dense model, rendering and compositing approaches. It was also necessary to find a way to estimate the RGBD camera's exposure and white balance settings.

The revised approach brought a number of benefits. The HDR information allowed for more accurate rendering, particularly of specular highlights on virtual objects. It also allowed the RGBD camera to enable auto exposure, capturing more pleasing real images to input into the MR result. It also removed the need for some manual adjustments to auto exposure and white balance that were necessary in the LDR version of the approach.

The new approach was tested in a variety of real environments, rendering a number of different virtual objects with diffuse and specular material properties. The results were of similar quality, with slight improvements in the quality of shadows cast by virtual objects, and better reflections of bright light sources. Removing the need for manual adjustments made getting good results using the application more straightforward.

Two different methods for updating the coarse model using bracketed exposures from the fisheye camera were developed and tested. In static scenes, the Kalman filter-based approach could reduce some of the temporal noise present in the input. Where there was motion in the real scene, the best-score approach responded more slowly and produced unnatural-looking artefacts. The Kalman filter-based approch responded more quickly, but suffered from artefacts more similar to motion blur. On the whole, the Kalman filter approach was considered more successful.

The approach improved on the earlier method presented in chapter 4 in a number of ways, but still suffered from some of its shortcomings. In particular, prior information about the real environment was still required, in the form of a floorplan, ceiling height and initial camera pose. There were also still restrictions on the possible shape of the real room, which was required to be a right prism due to the coarse model occlusion-handling method (see section 4.3.4). For this reason, in the final chapter of this thesis an alternative approach was explored, which removes the need for this prior information.

Chapter 6

Instant Illumination Capture

The content of this chapter comes from the following paper, which appeared in VRST 2018:

WALTON, D.R., STEED, A. Dynamic Environment Capture for Mixed Reality. In *24th ACM Symposium on Virtual Reality Software and Technology, VRST 2018* (2018)

6.1 Introduction

The systems presented in chapters 4 and 5 were capable of producing realistic results in the correct setting. However, they required some prior information about the scene, and placed some restrictions upon its geometry. Specifically, the system can only be used in an indoor setting, inside a room which is a right prism, and the geometry of this room must be known. In practice, this limits the range of applications for the system, which motivated the development of a newer, more general approach.

The new approach focused on removing these requirements, and producing a system capable of capturing illumination information in any environment, without the need for prior knowledge or real scene geometry. Focus was also placed on producing an approach which was more responsive to changes in the real environment.

6.1.1 System Overview

Similarly to the approaches discussed in chapters 4 and 5, a dense RGBD SLAM system was used to capture detailed real geometric information about the environ-

ment. Whilst the earlier approaches used the InfiniTAM volumetric approach, the new approach instead used a surfel-based SLAM system implemented by the authors. This system was based upon the system detailed by Keller et al. [74], but a number of significant modifications were made to better adapt it for use in a mixed reality system.

Firstly, HDR colour values were captured at each surfel. Similarly to the approach from chapter 5, the forward-facing camera has auto exposure enabled, and its exposure and white balance are inferred using the fisheye camera. The colours are converted to a linear colour space and stored at each surfel. A number of modifications were also made to the manner in which the surfels are updated with new RGBD data, enabling the surfel model to capture changes in the real environment more quickly.

Secondly, the fisheye camera is also used to update the colour values for surfels visible to it. This increases the number of surfels updated each frame, meaning the model updates to reflect changes in real-world lighting more quickly.

The earlier mixed reality approaches required rough geometry for the real room to be known a priori, in order to create a coarse model to be updated using data captured by the fisheye camera. In the new approach, this requirement is removed. Instead, the data captured by the fisheye camera is used to create HDR environment maps, which are intended to store the more distant real environment not captured by the RGBD reconstruction.

The main disadvantage of an environment map, as compared to the previous coarse model, is that the environment map can only capture incoming light at a single location in space. Consequently, if the user moves a significant distance from their starting location whilst using the application, the environment map will no longer be a reasonable representation of the lighting environment around them.

For this reason, the approach captures multiple environment maps in the form of cubemaps, each with an associated 3D location. These cubemap and 3D point pairs are referred to here as environment map keyframes. The 3D location indicates thefisheye camera position at the time the keyframe was created. When the camera

pair moves too far away from the location of the current environment map keyframe, a new one is created. Earlier keyframes are retained, and are used again if the camera returns to their location.

As with the earlier approaches, the environment map keyframes are inpainted to produce a plausible complete cubemap. Using the combination of an environment map keyframe and the dense model, the lighting environment around a virtual object can be estimated, and used to render it so that it blends convincingly into the real environment around it.

The new approach maintains the advantages of the previous approaches in chapters 4 and 5 - it is capable of generating a rough but complete model for a real environment very quickly, and then gradually improving the geometry over time, as more of the room is observed and reconstructed by the RGBD camera. It adds the ability to work in unprepared environments, widening the range of applications in which it can be used. The environment map model also opens up the possibility of use in unbounded outdoor environments, rather than enclosed rooms.

6.2 Approach Detail

This section describes each component of the improved approach in further detail.

6.2.1 Hardware

The hardware used for the new approach was identical to that used in chapter 5; a forward-facing Asus Xtion Pro RGBD camera, joined to an upward-facing camera Point Grey CM3-U3-31S4C-CS camera, fitted with a Lensagon CF5M1414 fisheye lens. The calibration was used again, and the same approach was used to undistort the fisheye images to a simpler camera model (see section 4.2.2 for more detail on the calibration).

6.2.2 HDR Surfel-based Reconstruction

As mentioned in section 6.1.1, the new approach used a surfel-based dense SLAM approach based on that of Keller et al. [74] to capture geometry near the virtual content and provide tracking. The newly adapted dense SLAM system, added the ability to capture HDR colours. The parts common to the SLAM system used here

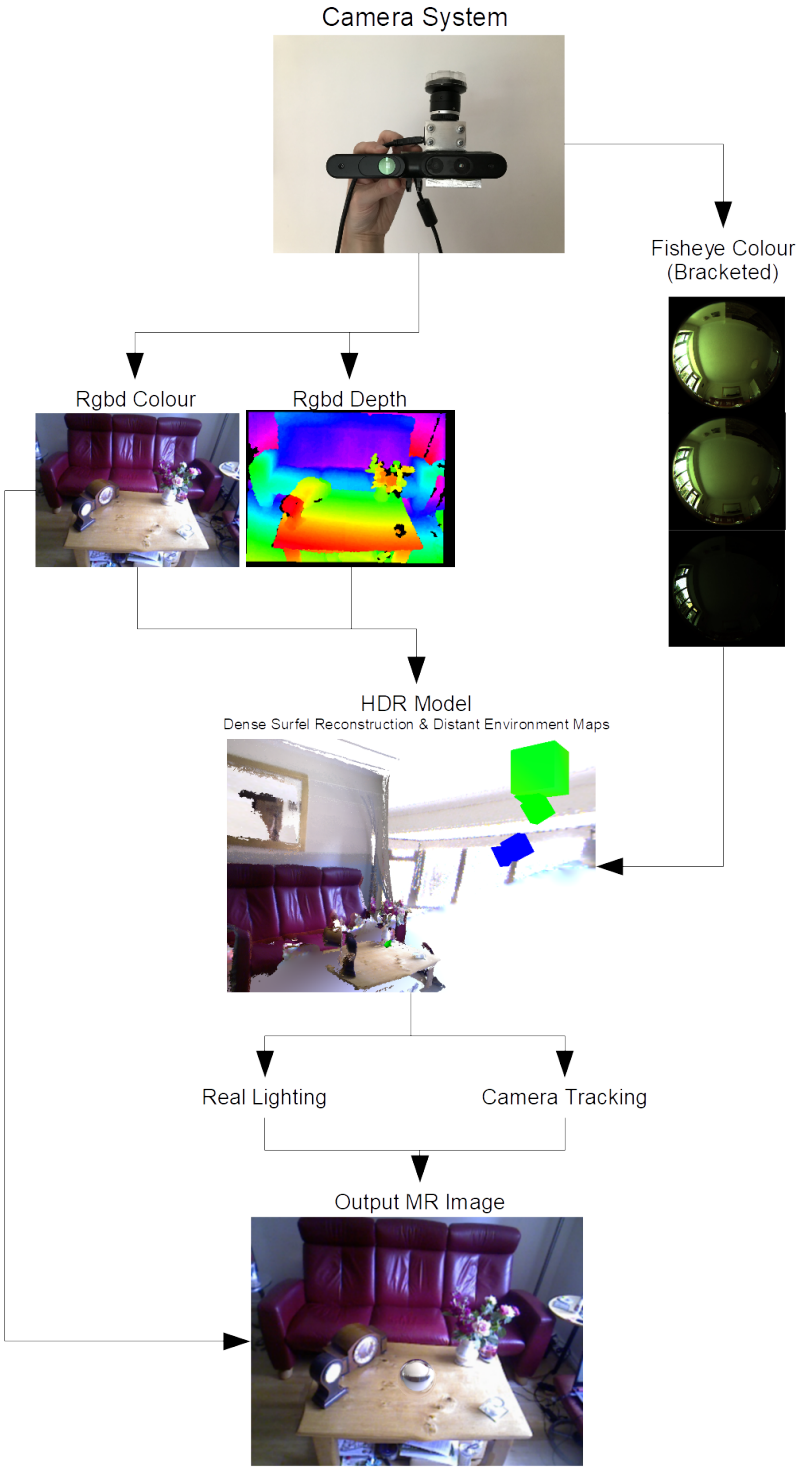


Figure 6.1: Flowchart giving an overview of the new approach

and the earlier system of [74] will briefly be outlined, and then the specific modifications made to capture HDR colour values will be detailed.

The model of the environment consisted of a list of surfels, each consisting of a point, normal and radius, along with a confidence counter and HDR colour value. The surfel list is initially empty. When a new frame is captured by the RGBD camera, it is projected to a number of surfels, each corresponding to a pixel in the depth map. At the first frame, these new surfels are simply added to the list.

On subsequent frames, the pose of the RGBD camera relative to the model is first determined using the camera tracking approach detailed in section 6.2.5. Following the tracking stage, the frame is again projected to a new set of surfels. Each new surfel can then either be added to the list, or used to update an existing surfel.

If a new surfel corresponds to an existing surfel (i.e. is close in 3D space, has a similar normal and radius) it is used to update the existing surfel. The update consists of taking a weighted average of the surfel's properties based on the existing surfel's confidence. This confidence value is incremented on each successful update. If the new surfel does not correspond to any existing surfels, it is added to the list of surfels.

Due to sensor noise, for example, "bad" surfels may be added via this process, that do not correspond to geometry in the real scene. For this reason, surfels below a confidence threshold are considered unstable, and not used for tracking. If a surfel remains unstable for a period of time, it is deleted from the list.

In order to quickly determine which new surfels correspond to existing ones, an index map is used. This is an image containing the existing surfels, rendered from the camera's viewpoint. Each is rendered into a 32-bit integer texture as a single pixel-width point, with the colour set to be the index of the surfel in the main surfel list. This map is consulted when updating the existing surfels, to quickly find possible correspondences. The index map is rendered at 4x the depth map resolution on each axis (i.e. 16x the total number of pixels) and each new surfel checks the corresponding 4x4 window for up to 16 possible existing surfels to find matches.

Before projecting a new set of surfels corresponding to the current frame, the observed LDR colour values are first converted to HDR values in a linear colour space. This was achieved in the same way as described in chapter 5. Briefly, inverse gamma correction is applied, and then the auto exposure and white balance are estimated by comparing to the images captured by the fisheye camera.

When converting the image to HDR, a flag is also stored for each input pixel, indicating if it was saturated or below the blackpoint in the input LDR image. This information is used to assign a lower initial confidence to these surfels, so their colours are replaced as quickly as possible with more accurate estimates.

6.2.3 Updating Surfel Colours

The new approach also includes a method for the fisheye camera to update the colours of surfels in the dense model of the environment. This allows their true intensity to be determined, in the event that they were saturated/black in the RGBD colour image used to create them initially, and also allows the intensity estimate to be refined. It also allows them to be updated in the event that the real lighting in the scene changes. In this way, the extra cameras extend the field of view, meaning that more of the dense model's colours can be updated each frame.

First, the pose of the camera must be determined. This is inferred from the pose of the RGBD camera (obtained through the tracking approach described in section 6.2.5) and the relative pose of the two cameras (obtained via camera calibration). Generally, the image will not have been captured at the same time as the most recent RGBD frame, so a simple linear interpolation is performed using the two most recent RGBD poses, to estimate the RGBD pose at the present time more accurately.

The updating uses an index map rendered using this camera pose, in a similar way to the RGBD updating process. When searching for possible matches in the 4x4 window, the match closest to the camera is selected, providing it satisfies a number of criteria:

1. The matched surfel lies in front of the camera, and is at least a minimum distance away (10cm).

2. The matched surfel is not too distant from the camera (over 10m).

3. The normal of the matched surfel points towards the camera (i.e. $c.n > 0$, where c is the vector from the camera to the surfel and n is the surfel normal).

4. The confidence of the surfel is above a threshold (5).

5. The colour of the matched surfel is sufficiently similar to that of the new surfel.

The first and third criteria make sure the match is actually visible to the camera. The second ensures that the surfel is not so far away that the resolution of the camera image will be insufficient to determine its colour. The final one is imposed because the location and normal of new surfels tends to vary a lot in the first few frames, meaning they may be matched incorrectly.

This updating process can also be applied when the camera used is omnidirectional, as in the case of the fisheye camera used in our prototype. In this case, an index cubemap is rendered instead of 2D index map image. The rest of the updating process is similar - for each 4x4 set of indices in the cubemap, a matching surfel is potentially found, and if so, its colour is updated using a sample from the omnidirectional image.

6.2.4 Environment Map Keyframes

In addition to the surfel-based component of the reconstruction, environment map keyframes were also captured. These each consisted of a cubemap and 3D location, as briefly mentioned in section 6.1.1. Similarly to the coarse model used in the earlier approaches, these were intended to capture details of the real environment not present in the dense model, either because they have not yet been observed, or they are too distant for their depths to be inferred by the RGBD camera.

The first keyframe is created at the initial location of the camera system. Its colours are initialised using the output of the fisheye camera, and then it is inpainted using an efficient GPU-based push-pull approach, slightly adapted from that in [47, 92]. This newer inpainting approach was significantly more efficient than that used in the previous approaches, enabling it to be performed on every frame.

The inpainting approach uses the labels assigned to each pixel, indicating if they have been observed by the fisheye camera. Those pixels that have been observed are not modified, and those that have not are replaced with inpainted data.

This inpainting approach takes place in two stages - the first "push" stage being similar to mipmap generation. Working from the finest level of the mipmap to the coarsest, each pixel is replaced with the average value of all valid parent pixels (i.e. all those which have been directly observed by the fisheye camera). In the second "pull" stage, working from coarsest to finest mipmap level, each pixel which has yet to be observed or inpainted is replaced with the colour stored in its parent pixel.

This procedure is carried out on each face of the cubemap. There is a potential issue, in that some of the faces of the mipmap may contain no observed pixels. Typically, for example, on the first frame the fisheye camera faces upwards, meaning that the bottom face of the cubemap will not yet have been observed. For this reason, a check is added to ensure that each face has at least one valid pixel before inpainting. If it does not, its neighbouring faces are inpainted first, and a one-pixel width boundary region is copied to allow the face to be inpainted successfully. Figure 6.2 shows an example of the result of this inpainting process.

This inpainting is repeated each time the keyframe is updated, to ensure the inpainted regions are consistent with the rest of the keyframe. The alpha channel of the texture stores a covariance value, which is used when updating the keyframe. Pixels which have been inpainted and not yet directly observed have the alpha value set to zero.

Keyframes are updated whenever a new colour fisheye image is captured. Each texel of the cubemap is projected into the new image, and it is sampled and used to update the colour. When projecting, only the rotational component of the camera's pose is used (i.e. the points in the map are considered to be at infinity).

The update is carried out using the Kalman filter based update detailed in chapter 5.3.4. The Kalman filter was slightly simplified for this new approach; instead of using separate covariance values for each channel, only one covariance value was used for each pixel. This was stored in the alpha channel of the cubemap, meaning

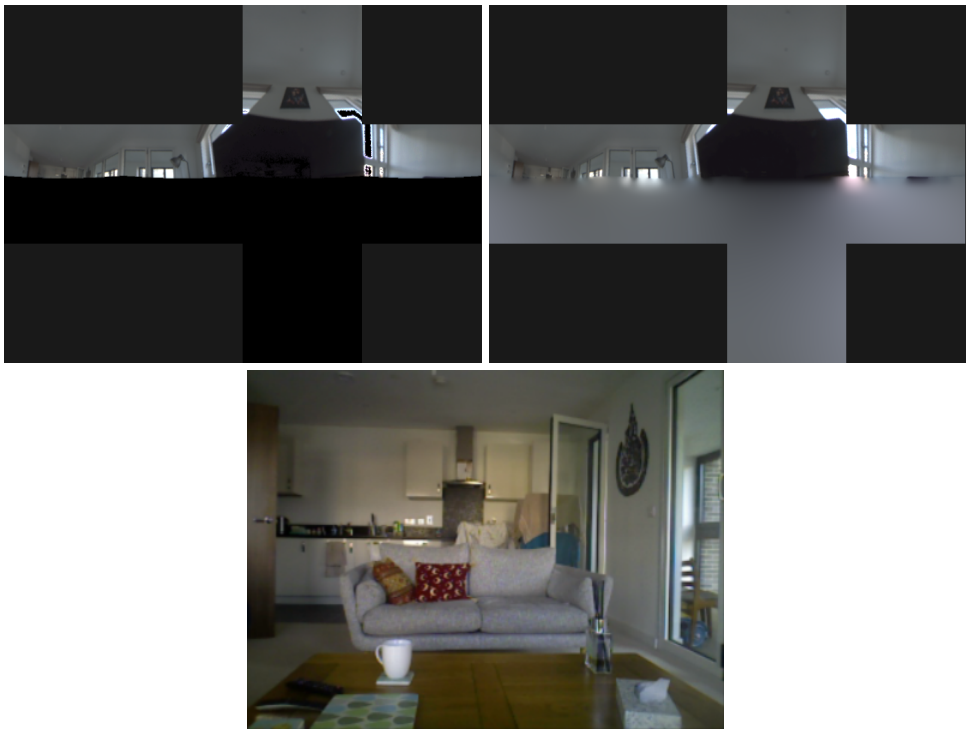


Figure 6.2: Environment map keyframes, before and after inpainting process. Top left: Environment map keyframe before inpainting. Black pixels in the lower half of the cubemap have not yet been observed. Some black pixels in the upper area of the cubemap were saturated or zero in all the fisheye images. Top right: Same environment map keyframe after inpainting. Bottom: Image taken from RGBD camera at the same time, for comparison.

that only one texture was required per environment map keyframe. This reduced the required GPU memory, which was important as this new approach could potentially generate several environment map keyframes. This simplification was not found to have a noticeable impact on the quality of the results.

If the pixel to be updated is being observed for the first time (i.e. has an alpha value of zero), however, it is simply replaced with the observed colour. If the observed colour is saturated or black, no update is carried out, as the true radiance at this pixel is unknown.

A new keyframe is generated when the translational component of the camera system's pose exceeds a threshold. This helps to prevent the keyframe becoming obsolete when the distant environment assumption is violated. New keyframes are initialised with the colours from the previous keyframe, and the alpha channel set to zero (indicating the pixels have not yet been observed). Previous keyframes are

retained, and if the camera moves back within range of an existing keyframe, that keyframe will subsequently be updated and used in rendering.

When new surfels are added, the environment keyframe can be consulted to obtain HDR colours for them, where available. This is done when a newly added surfel is either saturated or below the black point in the RGBD image, meaning its intensity would otherwise be unknown. The surfel is projected back into the environment map keyframe to find a colour, and this colour is used if the alpha channel is non-zero (i.e. this part of the keyframe has been observed by a camera, and was not inpainted or propagated from a previous keyframe).

6.2.5 Camera Tracking

Similar to other dense RGBD SLAM approaches, the location of the RGBD camera is tracked via an iterative dense frame-to-model alignment process as each new frame is observed. The camera tracking implementation used in this implementation made use of the colour and depth tracking approach of Kerl et al. [75], adapted to make use of the HDR information stored in the surfels.

Specifically, the RGBD alignment process developed by [75, 76, 131] was used. Given two pairs of intensity and depth images, this finds the rigid transform which best aligns them, in the sense that it minimises the colour and intensity differences.

This was used as part of a similar overall frame-to-model tracking approach to earlier dense RGBD reconstruction methods including Kinect Fusion [106]. When each new pair of colour and depth images is captured by the RGBD camera, the current surfel model was rendered from the previous RGBD camera pose. The newly observed frames were then aligned to these rendered frames. This provided an estimated transform for the current frame, which was composed with the previous camera pose estimate to provide an updated camera pose.

The HDR information available in the model was used to improve the tracking performance. When rendering the colour frame, the current estimated exposure and white balance settings of the RGBD colour camera were applied. These were determined as outlined in section 6.2.2. This ensured that the effective exposure

and white balance in the real and rendered colour images was the same. The colour images are then converted to grayscale intensity values, and the intensity and depth images aligned using the approach of Kerl et al. to provide the final transform.

An alternative approach would have involved converting the input RGBD image to HDR, and aligning this to a HDR rendering of the dense model. This possibility was explored during development of the approach, but did not perform as well in cases where the input image contained saturated or zero-value pixels. If LDR images are aligned, these pixels will be saturated or zero in both the real and rendered images, and the information in these pixels can be used to align the images. However, if HDR images are used in tracking, these saturated or zero pixels are unable to be converted to HDR and cannot be used in tracking.

Tracking using HDR colours and depths brings a number of advantages over earlier approaches such as Kinect Fusion [106], which tracks using depth alone. Should the depth image lack sufficient geometric detail to enable the tracking to function (e.g. the camera observes a flat wall), the colour tracker is still often able to successfully produce an estimate. Conversely, if colour tracking fails (for example, due to a change in lighting conditions), the depth tracking is often able to successfully recover a camera transform. Since we track using HDR colours, rather than the LDR colours used by InfiniTAM [61], for example, it is also possible to enable the auto exposure and white balance features of the colour camera without adversely affecting the colour tracking. This allows the camera to capture a greater number of correctly exposed pixels, and to capture a more pleasing image to present to the end user.

6.2.6 Dynamic Geometry

Early dense RGBD reconstruction approaches, including KinectFusion [106] focused on reconstructing an accurate model of a static real environment, and contain measures to avoid integrating dynamic moving objects into the reconstruction. However, the intention of this approach was to respond in real time to changes in geometry, enabling virtual content in MR applications to (for example) reflect dynamic real geometry, such as the user's hand. To achieve this, we adapt and build

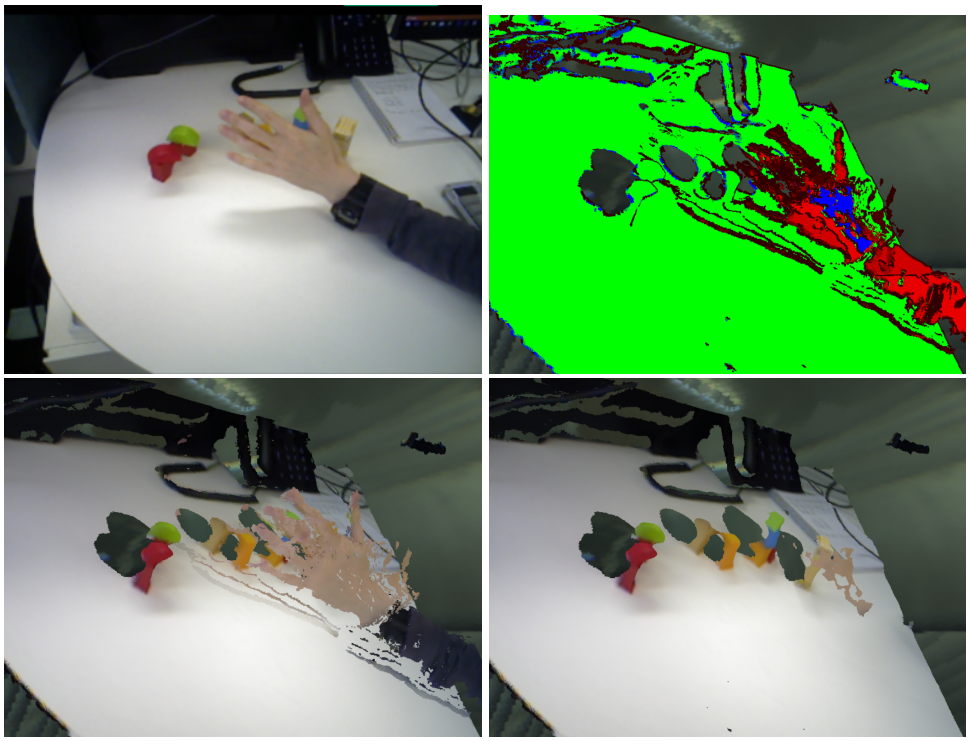


Figure 6.3: Examples of surfels used for tracking vs. surfels used for rendering MR content

upon the method of Keller et al. [74], which explicitly detects pixels corresponding to dynamic moving objects, and handles them separately.

Similarly to Keller et al., the criteria used when using surfels for tracking and MR are separated. Only those surfels that qualify as stable (based on the confidence threshold detailed in section 6.2.2) are used for tracking, as in [74]. However, all surfels are used when rendering for MR (for example, when rendering environment maps used to simulate reflection from a virtual object). This enables the MR rendering to respond instantly to geometric changes, without impacting the quality of the camera tracking.

Figure 6.3 shows an example of the difference between these categories. The top image is the input colour image from the RGBD camera. The bottom left image shows a rendering of all the surfels captured by the approach. The bottom right image shows just those surfels which are categorised as stable. The top right image shows a visualisation of the confidence values of the surfels: here shades of red are unstable, and stable surfels range in colour from blue to green as confidence increases. Here, the hand has just moved into the frame. It is present in the surfel

model, and so will correctly impact the lighting of virtual objects, but is not present in the set used for tracking, so will not negatively impact the tracking.

In many reconstruction methods including [106], colours are generally updated using a rolling-average approach, similar to that used by other approaches such as and . This approach is helpful in eliminating noise present in the colour images by averaging the colours of multiple observations of each surfel. However, Keller et al. [74] simply store the most recent colour observation at each surfel, which results in much quicker colour updating at the cost of extra noise. We use the rolling-average approach, but where a newly observed colour differs significantly from the existing surfel, its colour is instantly replaced with the new value and its confidence is reduced. This brings the benefits of both approaches; the averaging reduces noise, but when, for example, a light is turned on or a window opened, our approach can respond to this change quickly and remain consistent with the real world.

In order to respond quickly to dynamic objects, it is also critical to quickly remove surfels which no longer correspond to valid real geometry. This allows the model to better reflect reality in the event that a real object is moved or removed. [74] implement one method for achieving this, which we also employ. This involves searching for those surfels which have a smaller depth than that observed in the current frame. Since we assume surfels are opaque, this is a physically impossible situation, and indicates that these surfels should be deleted from the model. These surfels are identified and removed during the updating stage, using the index map.

In some situations, this method fails to remove bad surfels. This is a particular problem when depth values are missing, or when depth is insufficient to identify the problem (e.g. a piece of paper is moved along a tabletop). Since depth values frequently cannot be obtained, small clusters of erroneous surfels are often left behind in the model.

In this approach, the problem is addressed by also checking for colour violations. These are points where the colour of the visible surface of the model differs from that actually observed by the camera. These are identified by rendering a view of the model from the current camera pose, and comparing the HDR colours with

Figure 6.4: Surfel reconstructions after sliding a real piece of coloured paper across a table.

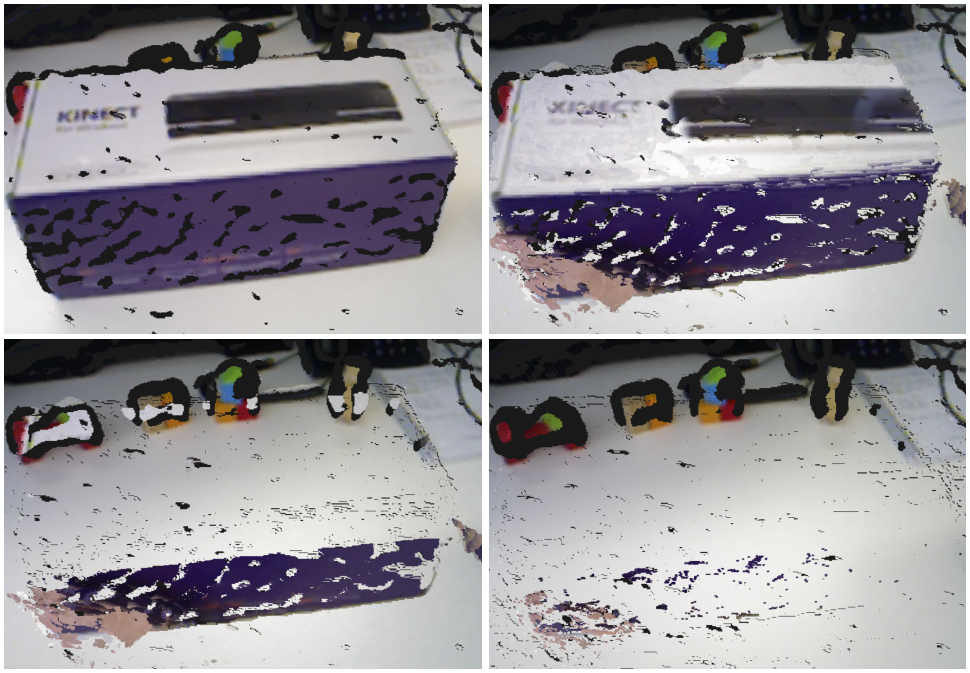


Figure 6.5: Surfel reconstructions before and after removing a box from the real scene.

the real image at each pixel. At pixels where the colour is sufficiently different, a colour violation is detected and the corresponding surfel is marked for deletion.

In order to improve efficiency and avoid removing surfels erroneously, the colour violation check is only carried out at pixels where no valid depth values could be obtained. This is based on the assumption that where valid depth values are available, bad surfels would be detected as depth violations, so carrying out the colour violation check is unnecessary. This makes the violation check more efficient, and helps to avoid erroneously removing surfels.

Figures 6.4 and 6.5 show examples of how each of these approaches improves

the updating of the surfel model. Figure 6.4 shows the surfel reconstruction after a green piece of paper has been moved from left to right over a tabletop. In the left example, standard colour updating was used, and the colours are slowly updating, leaving an afterimage of the paper's previous position. In the right example, our faster colour updating approach was used and the colours were updated instantly, removing this artefact.

Figure 6.5 shows the importance of removing depth and colour violations from the surfel model. The top left image shows the initial surfel reconstruction of a box placed on a table. The other images show the state of the reconstruction a few frames after the box has been removed. In the top right example, neither color nor depth violation removal was used, and the surfels corresponding to the box remain in the model. In the bottom left example, just depth violations were removed. Much of the box is gone from the model, but the lower half had depths similar to those of the table below, so these could not be classified as depth violations and removed. In the bottom right, removing both depth and colour violations has successfully removed nearly all of the outdated surfels corresponding to the box from the model.

6.2.7 Rendering Virtual Content

The MR rendering and compositing approach used here was very similar to that employed in chapter 5. Briefly, in order to render a virtual object, a HDR environment map was first rendered from the virtual object's location. The closest environment map keyframe to the virtual object was rendered into this environment map first, followed by rendering the dense surfel-based reconstruction from the virtual objects location. This environment map was used to render the virtual content into a floating point HDR texture, which was composited with the HDR-converted image from the RGBD camera. The result was then converted back to LDR for display.

When rendering the surfel model, backface culling was disabled, as it is common for only the side of an object facing away from the virtual content to be observed by the camera system. Rendering backfaces ensures that the object will be present in reflections in the virtual content, although the geometry and colour of the real object may be inaccurate. Using the surfel-based approach improved the

efficiency of rendering the model, as the rendering step simply involved rendering surfels using a geometry shader rather than an expensive volumetric ray-casting operation.

One issue commonly encountered with the previous approaches was that of small holes in the dense reconstruction. These were common when surfaces had only been observed at steep angles. These small holes would become particularly noticeable when a reflective object was placed on a flat surface, such as a tabletop. The holes would be magnified due to the close proximity of the virtual object to the surface, and then would be very visible in reflections on the lower half of the object. The issue can clearly be seen in figures 4.11 and 4.12.

Although the gaps in the model proved to be smaller using the surfel-based approach, they were still noticeable. For this reason, a post-processing step was added to fill these small gaps between the rendered surfels in the environment maps. After rendering the dense content, a simple inpainting step was performed on the GPU. This performed a morphological dilation operation on the rendered dense reconstruction, followed by an erosion operation. This had the effect of filling these small gaps between surfels efficiently. The alpha channel of the cubemap texture was used to identify which pixels were rendered from the dense reconstruction, and which were from the environment map keyframe. An example of the effect of this inpainting procedure is show in figure 6.6.

Shadows cast by virtual objects and diffuse virtual objects were again rendered using differential PRT, as in the previous approaches in chapters 4 and 5. However, the transfer coefficient baking stage was improved from that used in chapter 5 in the case of shadows cast by specular virtual objects. Previously, these were handled using diffuse PRT, and specular virtual objects were treated the same way as diffuse ones for the purpose of differential rendering. This lead to shiny virtual objects casting excessively dark shadows.

In the new approach, during each light bounce in the baking stage, if a ray hits a specular virtual object, it is reflected in the object and cast out into the scene again. This effectively simulates a virtual object with an ideal mirror surface, and allows

Figure 6.6: Effect of inpainting the dense model in the output environment map. Top: MR scene, with virtual trumpet in real room. Bottom left: environment map for trumpet, without inpainting. Note the incomplete regions on the table and on the back walls. Bottom right: environment map with inpainting. Note these missing regions have now been inpainted.

light reflected off the virtual object onto a real object to contribute to the lighting simulation. This was intended to improve the accuracy of shadows cast by shiny virtual objects and to open up the possibility of simulating effects such as caustics.

6.3 Results

The application described in section 6.2.7 was tested in a variety of real scenarios, with a number of different virtual objects.

6.3.1 Application Results

Figure 6.7 shows examples of a number of different virtual objects placed in different real scenes, demonstrating some of the virtual materials which can be effectively simulated using the approach. These include reflective surfaces with surface colour and roughness, as well as diffuse surfaces with albedo textures.

In contrast to the earlier approaches detailed in chapters 4 and 5, no information

Figure 6.7: Examples of MR frames rendered using the approach, containing a variety of different virtual objects and real scenes. Top left: virtual pan, with ideal (mirror) reflection. Top right: virtual teapot, with slightly rough, coloured reflective surface. Bottom left: virtual terracotta bunny, with diffuse surface. Bottom right: virtual trumpet, with rough coloured reflective surface.

about the real scene had to be captured beforehand in order to obtain these results. The application was simply started in the real environment, and the pose of the virtual object was then set to place it correctly on top of the real surface.

6.3.2 Environments with Wide Dynamic Range

Figure 6.8 shows an example where the system was used in a scene with a wide dynamic range, demonstrating the ability of the approach to function correctly in such scenes. This scene was captured in a flat on a sunny day, and the windows are significantly brighter than the rest of the scene, however the bracketing approach outlined in section 6.2.1 enables the full dynamic range of the scene to be captured. Additionally, as the camera is moved around the real table, its auto exposure adjusts due to the bright windows. This change in auto exposure is detected, and the brightness of the virtual bunny adjusted appropriately when mapping the output to LDR. This means the bunny appears darker, and remains consistent with the real scene

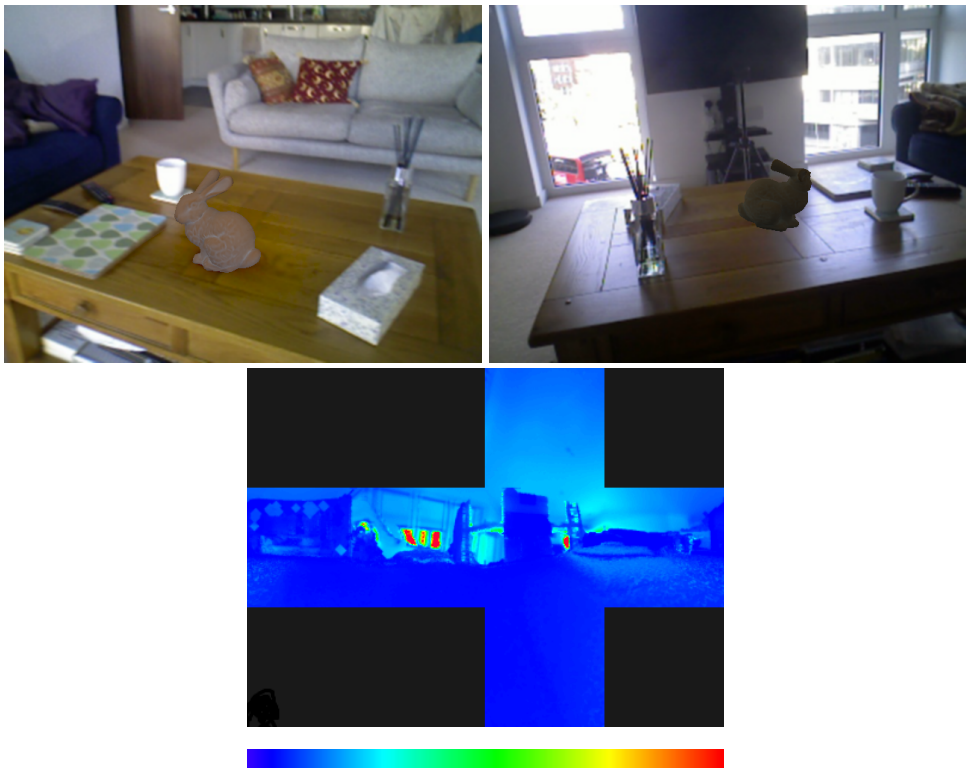


Figure 6.8: Example using the system in an environment with a wide dynamic range. Top left: view of virtual bunny. Top right: view of virtual bunny from other side of table. Note that the auto exposure has adjusted to compensate for the bright windows, but the bunny remains consistent with the real scene. Bottom: HDR radiance environment map used to render bunny, shown as heatmap (colour key below). Note that the windows are much brighter than the rest of the scene.

around it.

6.3.3 Comparison to Physical Light Probe

In order to assess the realism of the results produced by the application, a similar qualitative comparison to that carried out in chapter 4 was performed. Again, a real chrome sphere was compared to a virtual sphere of the same size rendered by the application.

The virtual and real spheres appear similar overall, with nearby reflected objects located correctly in the virtual reflective sphere, and of similar appearance. There is some difference in the colour of the two spheres - in particular, the real sphere is somewhat darker. This is due mainly to the fact that the virtual sphere was rendered with an ideal mirror surface, wheras the real sphere is an imperfect reflector. Additionally, the real sphere has a sharper shadow than the virtual sphere.

Figure 6.9: Comparison of a virtual reflective sphere rendered using the application with a real chrome sphere of the same size. Above: Real sphere. Below: virtual sphere.

This is due to the shadow using PRT, with only 4 SH bands, and consequently being unable to capture the higher frequencies of the incident lighting.

There are slight differences in the positions of nearby real objects due to the use of environment mapping. The reflections on the sphere were simulated using an environment map rendered from the centre of the sphere, which results in slight geometric inaccuracies when this map is used to simulate reflections on the surface of the sphere.

The most noticeable difference between the two spheres is in the positioning of the distant geometry. This is slightly different due to being rendered using the environment map keyframe, which was captured at a different point in space (i.e. at the fisheye camera location). The earlier approaches of chapters 4 and 5 addressed this problem using the coarse model geometry. This improved the accuracy of the placement of the distant geometry, but this came at the cost of requiring more geometric information a priori, in addition to limiting the range of possible real environments in which they could be used. This approach offers greater flexibility and ease of use at the cost of this slight decrease in accuracy. Although the geometric inaccuracy

is noticeable when directly compared to a real sphere, the issue is less noticeable in practice and the results are perceptually plausible, if less accurate in an absolute sense.

6.3.4 Comparison to Prior Approaches

The comparison from section 5.4.3 was extended to compare the output of this new approach with the earlier approaches of chapters 4 and 5. Again, since the three approaches accept different inputs (bracketed HDR input vs LDR input), it was not possible to record a single video and use it as input to all the approaches. In order to provide the fairest comparison possible, in each of the following examples the footage was captured for each approaches one after the other, without any change to the real scene. In each case, the camera was moved around the scene briefly to generate a satisfactory reconstruction, before being clipped into a sturdy quick-release tripod. This tripod remained in the same position throughout, so that at the end of each sequence the camera had the same view of the real scene. These recorded sequences were then processed using each of the approaches, and the final frames captured from the same viewpoint are shown in figure 6.10 for comparison.

There are some slight geometric inaccuracies common to all of the approaches - for example, the reflection of the monitor behind the sphere is slightly misplaced, and too large. This is mainly a consequence of the use of environment mapping, which is common to all three approaches. The reflections are all simulated using an environment map rendered from the centre of the virtual sphere, and this results in slight inaccuracies when this map is used to simulate reflection from other points in space, such as the top of the sphere. In principle, this issue could be corrected in any of the three approaches by changing the rendering method to use another approach, such as ray-tracing.

The new, instant illumination approach also has slightly incorrect placement of more distant parts of the environment, such as the overhead lights. This is a consequence of the use of environment map keyframes, which assume a distant real environment. The two approaches using a coarse model are able to achieve more accuracy in this regard, at the cost of requiring real environment information a pri-

Figure 6.10: Comparison of a virtual reflective sphere rendered using the three systems discussed in this thesis with a real chrome sphere of the same size. Top: Real sphere. Top centre: Newest approach from chapter 6. Bottom centre: Original LDR approach from chapter 4. Bottom: HDR approach from chapter 5.

ori. In practice, whilst these inaccuracies are apparent when the output of the instant illumination approach is compared to ground truth, the reflections in the virtual object appear perceptually plausible, which is sufficient for many applications.

The dense model in the newer approach was generated using the reconstruction method detailed here, which was intended to prioritise rapid updating over detail and accuracy. This has resulted in in less detail in the reflections of the pens and eraser placed on the table in front of the sphere, as compared to the earlier approaches. On the other hand, the dense model inpainting process added to the newer approach has removed the artefacts caused by small holes in the dense reconstruction. In the earlier approaches, these artefacts are visible as small patches of incorrect colour in the reflection of the table below the sphere.

6.3.5 Limitations

This approach produces plausible results in a number of settings, but some limitations of the approach were discovered during testing, which will be discussed here.

Since the approach relies upon the dense surfel reconstruction and environment map keyframes, it may produce inaccurate results when these are incomplete. This can be problematic when the depth camera cannot infer depths for some objects in the real scene, for example if a real object is translucent or highly specular. Such objects will not be present in the reconstruction, and will therefore not be present in reflections from virtual objects added to the scene. This problem is common to most dense reconstruction approaches using RGBD cameras, and is not easily addressed with current depth sensing technology.

The colour violation removal approach detailed in section 6.2.6 can sometimes erroneously remove valid surfels from the reconstruction. Small numbers of surfels can be removed due to slight tracking inaccuracies or non-lambertian real objects (e.g. specular highlights). The problem is most noticeable, however, when a real object moves very close to the RGBD camera. Figure 6.11 shows an example of this issue, where a user's hand has moved very close to the RGBD camera. Here, the camera was unable to find depth values for the hand since it fell outside the depth range of the camera. As a result, all the surfels behind the hand are labelled

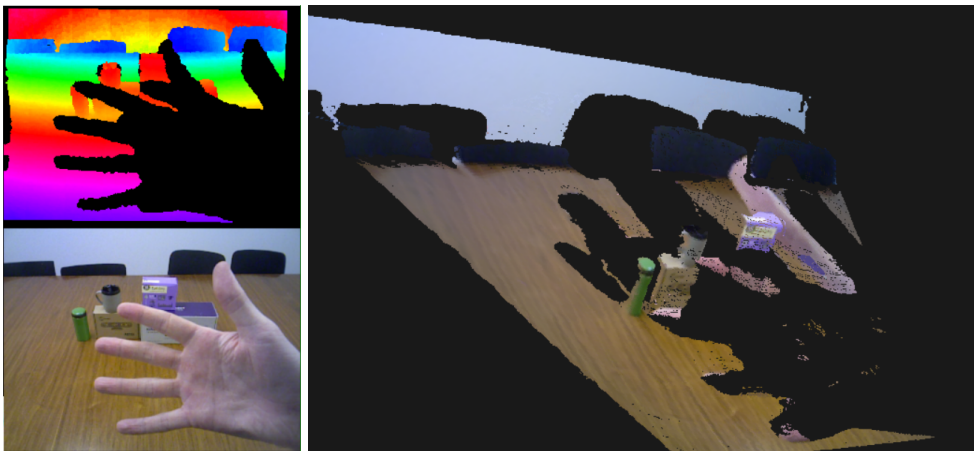


Figure 6.11: Examples of a failure case in the surfel reconstrution. Surfels have been incorrectly removed from the model when a user placed their hand too close to the RGBD camera. Bottom left: depth and colour inputs. Bottom right: rendering of surfel reconstruction.

as colour violations and removed. This problem would be partly mitigated by using a more recent RGBD camera with a lower minimum depth.

As mentioned in section 6.2.7, the PRT shadow rendering for specular objects was updated to simulate specular reflection from the object. However, since the object was modelled as a perfect mirror surface, this tended to result in shadows which were not sufficiently dark. This issue can be seen in the comparison with the real sphere in figure 6.9. This issue could be addressed in future work by instead using a more accurate surface model which absorbs or scatters a proportion of the incoming light.

6.4 Conclusion

In this chapter, a new approach to probeless illumination capture was presented and explored. The approach focused on removing some of the restrictions on real environment and prior information requirements of the earlier approaches presented in chapters 4 and 5, whilst maintaining the advantages of these earlier approaches as far as possible.

Through testing, it was shown that the new approach produced results of similar quality to these earlier approaches, and the new approach could be used more easily in a wider range of real environments. The newer approach was also shown to

be capable of responding more quickly to dynamic changes in the real environment, thanks to a specially adapted dense reconstruction approach.

The new approach did suffer from a slight loss in geometric accuracy with respect to the reflections of distant environment objects, as compared to the approaches from chapters 4 and 5 (see section 6.3.3). Although the results produced by the new approach are still perceptually plausible, this might mean that there are some application areas where these earlier approaches may be better suited. Specifically, if an MR system is to be used in a particular indoor environment where the floorplan and initial camera pose are known a priori, the approach of chapter 5 may produce slightly better results. If a system is to be used in applications where these requirements are not met, however, the new approach would be more applicable.

Chapter 7

Conclusions

7.1 Overall Summary

The goal of this project was to explore methods to improve the rendering of virtual content in a video see-through MR setting. We focused on methods that could be used in a variety of environments, and that could produce results in real time, without requiring precapture or precomputation steps. The methods we developed made use of modern camera technology to capture as much information as possible from the real environment.

During the EngD project, a number of approaches were developed to tackle two related problems in this area. The first was that of handling occlusion of virtual objects by real ones, using the incomplete and noisy geometric information provided by an RGBD camera. The second problem was capturing dynamic, high-frequency lighting information for rendering virtual objects, without the need to place physical light probes at the virtual object locations.

7.2 Occlusion in MR

The first part of the project investigated how the noisy and incomplete depth and colour output of an RGBD camera could be used to correctly handle occlusion of virtual objects by real ones in an MR scene. The methods developed differed from previous approaches in the literature in that they were purely image-based, and did not rely upon reconstruction of the real environment. This meant that the approaches were ideal for handling occlusion of virtual objects by real ones which are

difficult or impossible to reconstruct using current reconstruction techniques, such as dynamic, non-rigid real objects.

A number of different occlusion-handling approaches were explored. The approach based on Cost Volume Filtering was found to be the most promising, and this was compared quantitatively with other approaches from the literature. In order to perform a meaningful quantitative comparison against dense RGBD SLAM approaches, a new comparison method was developed. This comparison method was used to demonstrate that CVF occlusion could produce better results than other frame-by-frame approaches. The results were also comparable to those produced by InfiniTAM, at much lower computational cost. CVF occlusion had higher temporal noise than InfiniTAM, but was capable of handling dynamic real objects.

7.2.1 Future Works

Since the paper on the work detailed in chapter 3 was submitted, there has been further work in the area. The most directly related publication was that of [53], which details a different approach using the same inputs. As more work is carried out in this area in the future, techniques for quantitavely comparing the qualities of competing techniques, such as that detailed in section 3.3 will become increasingly valuable.

Consumer RGBD hardware is also gradually improving, with newer cameras offering better depth sensing capabilities. Cameras such as the Intel Realsense D435 camera offer higher output resolution and greater depth range than the Asus Xtion Pro camera mainly used in this EngD project. The newer Intel camera is also capable of operating outdoors in sunlight, albeit at the cost of noisier depth output. These newer cameras could enable increasing the quality of the results produced by approaches such as those in chapter 3, in addition to extending the possible range of scenarios in which they could be used.

Thanks in part to a growing range of available RGBD datasets [152, 58, 17, 25, 95], machine learning techniques using RGBD input are recently showing promise for a range of applications. Zhang et al. [159] recently proposed a technique for depth completion and denoising using a deep learning approach. Similar tech-

niques may prove fruitful in handling MR occlusion using RGBD data in the future, and could provide superior results to the current filtering approaches by using prior learned information (e.g. inferring the proximity of an object based on its perceived size).

RGBD sensors are naturally not the only way to recover depth and colour image pairs. RGB stereo techniques can also provide similar output, and it would be interesting to investigate applying occlusion handling techniques to RGB stereo output. However, the nature of RGB stereo output is quite different, with different accuracy and precision characteristics. RGB stereo and RGBD cameras both suffer from missing depth values, but typically at different image locations. It is likely that the techniques developed here for RGBD output would need to be adapted significantly to produce good results given RGB stereo data.

7.3 Real Environment Capture

The second part of the thesis investigated how detailed, dynamic, high-frequency lighting information about a real environment could be captured using a self-contained device, without the use of external light probes. It also explored how this information could be used in an MR application, to add virtual content which appears to be illuminated by the real environment around it.

A number of approaches were developed, all using a similar hardware setup with a forward-facing RGBD camera connected to an upward-facing camera with a wide field of view lens. An initial, LDR approach was developed first (chapter 4). This used the RGBD output to construct a dense colour reconstruction of the real environment near the virtual object, and the fisheye output to generate a texture for a geometry proxy for the more distant parts of the indoor environment (i.e. walls, ceiling and floor). Both parts of the model were then used to render realistic virtual content.

The second approach built upon the first (chapter 5). The main change to this new approach was to add the capability to capture HDR colour information about the real environment, and make use of this information when rendering the virtual

content. This was achieved using an exposure bracketing approach, and by adapting the model updating processes to handle HDR data.

Finally, an approach was developed which focused on removing the requirement for prior information about the real environment (chapter 6). Rather than using a geometry proxy, this approach captured a series of environment map keyframes. This new approach also captured HDR data, but was now capable of functioning in new, unprepared environments. A number of other improvements were made, including making the dense SLAM approach more responsive to changes in the real scene.

These approaches were evaluated qualitatively, via testing with a range of different real environments and virtual objects. They were also compared to one another and against ground truth by comparing a real chrome sphere to a virtual mirror sphere of the same size.

The initial LDR approach had a number of advantages over other probeless illumination capture approaches in the literature. Its was capable of capturing high-frequency illumination information and updating it dynamically in real time, enabling specular virtual objects to reflect their real surroundings convincingly. This was possible even when the real surroundings changed in a variety of ways - for example the image changing on a television screen, ceiling lights being turned on or off, or curtains being opened or closed.

The approach of chapter 5 added the capability to capture a HDR illumination model, allowing for more accurate lighting. It was capable of capturing the full dynamic range of the scene, and also brought other benefits. The auto exposure and white balance of the real camera could be enabled, and manually tweaking parameters such as the fisheye exposure and white balance and the virtual shadow intensity was no longer required.

The final approach of chapter 6 offered faster response to changes in the nearby real environment, thanks to the new RGBD reconstruction approach. It also removed the requirement for prior information about the real scene, increasing the range of applications for the approach. However, this came at the cost of slightly

decreased geometric accuracy in the reflections of the distant environment. The new dynamic RGBD reconstruction approach was also found to offer slightly less accuracy and detail in the captured model. For this reason, there may be a limited number of applications where the earlier approach of chapter 5 would serve better.

7.3.1 Future Works

The work carried out during this thesis focused mainly on ways to capture a colour model of the real environment suitable for MR applications, and less focus was placed on rendering methods using this model. Whilst a basic rendering approach using environment mapping and PRT was implemented, it would be interesting in the future to use these techniques with other real-time rendering approaches. At the time of writing, consumer GPUs with hardware-accelerated ray-tracing capability have recently been released, and these could be used to make excellent use of the detailed information present in the captured models.

The techniques detailed in chapters 4, 5 and 6 all assumed that real surfaces were Lambertian and grey for the purposes of differential rendering. This produced generally plausible results, but this simplification sometimes caused artefacts - perhaps most notably, virtual objects did not interact correctly with specular highlights on nearby real surfaces. It would be interesting in the future to explore how the differential rendering results could be improved by estimating real material properties. This could potentially be achieved by using a machine learning approach such as that developed by Meka et al. [97]. This would open up a wider range of realistic lighting interactions between the virtual and real objects in the scene.

7.4 Conclusion

The approaches detailed in this thesis have focused on capturing real environment information in order to enable rendering of more realistic-looking mixed reality content. In the taxonomy of Milgram and Kishino [98], the goal has been to enable approaches to move along the extent of world knowledge (EWK) axis, enabling richer MR interactions in unprepared environments.

Chapter 3 focused on the problem of correctly simulating occlusion of virtual

objects by real ones. Work in this area continues to develop. In addition to new approaches such as that of Hebborn et al. [53], depth devices continue to improve in quality, in addition to becoming cheaper and more compact. Devices such as the most recent Intel RealSense cameras[1] are also capable of being used in outdoor settings with brighter ambient light, increasing the range of settings in which they can be used. The author feels that in future MR systems, the real-virtual occlusion problem is likely to be solved using a combination of advances in sensing hardware and processing algorithms, possibly making use of new machine-learning methods.

Chapters 4, 5 and 6 detailed methods for capturing geometric and lighting models of real environments, and demonstrated a way in which these data could be used to render realistic virtual content into MR scenes. A major missing element of the real environment which these approaches were not yet able to capture was real material information. Obtaining this information would not only enable more realistic differential rendering, but would also open up the possibility of XR interactions. These could include capturing virtual replicas of real objects, or capturing and sharing a space as part of an MR telepresence application.

It is likely that future real material information estimation approaches will also use a combination of novel algorithms and novel sensing technology. For example, light field cameras can capture multiple views of a surface point simultaneously, which could provide useful extra data to infer its material properties. At the time of writing, light field cameras such as those developed by Raytrix[2] are expensive and targeted towards industrial applications. If, however, their usefulness for MR were demonstrated, it is possible that compact, cheap consumer cameras may become available.

In addition to geometry, materials and lighting, there is much more world knowledge which may be captured in order to enable richer mixed reality experiences. Capturing semantic information about the real environment would enable future applications to be more context-aware, understanding the purpose of real objects and how virtual content should interact with them. Approaches such as Seman-

[1] https://realsense.intel.com/
[2] https://raytrix.de/

ticFusion [95] show one way in which this could be achieved, and are an important step towards the goal of enabling compelling MR in unknown environments.

It is an exciting time for mixed reality at present, as many of the core technologies required to advance the field are converging. New techniques such as those described above are improving EWK, and simultaneously MR displays are becoming increasingly capable and affordable. MR now has the potential to become a more integral and important part of peoples' everyday lives.

Appendix A

Publications

This project has resulted in the following publications, appearing in peer-reviewed conferences:

WALTON, D.R., THOMAS, D., STEED, A., SUGIMOTO, A. Synthesis of Environment Maps for Mixed Reality. In *16th International Symposium on Mixed and Augmented Reality, ISMAR 2017* (2017)
Contains extracts of work presented in Chapter 4.

WALTON, D.R., STEED, A. Accurate Real-time Occlusion for Mixed Reality. In *23rd ACM Symposium on Virtual Reality Software and Technology, VRST 2017* (2017)
Contains extracts of work presented in Chapter 3.

WALTON, D.R., STEED, A. Dynamic Environment Capture for Mixed Reality. In *24th ACM Symposium on Virtual Reality Software and Technology, VRST 2018* (2018)
Contains extracts of work presented in Chapter 6.

Appendix B

List of Acronyms

3D Three Dimensional

AR Augmented Reality

AV Augmented Virtuality

CPU Central Processing Unit

CVF Cost Volume Filtering

DoF Degree(s) of Freedom

DSO Direct Sparse Odometry

EWK Extent of World Knowledge

GI Global Illumination

GNSS Global Navigation Satellite System

GPU Graphical Processing Unit

HDR High Dynamic Range

HMD Head-Mounted Display

LDR Low Dynamic Range

LI Local Illumination

LIDAR LIght Detection And Ranging

LSD-SLAM Large Scale Dense SLAM

LUT Look-Up Table

MR Mixed Reality

MSE Mean Squared Error

PRT Precomputed Radiance Transfer

PTAM Parallel Tracking And Mapping

RADAR RAdio Detection And Ranging

RGB Red, Green and Blue

RGBD RGB and Depth

SDK Software Development Kit

SAR Spatial AR

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localisation And Mapping

SH Spherical Harmonic(s)

XR Cross Reality (also Extended Reality)

Appendix C

Colophon

This document was set using LaTeX and BibTeX using the UCL Thesis document class, composed with vim.

Bibliography

[1] Hassan Afzal, Kassem Al Ismaeil, Djamila Aouada, François Destelle, Bruno Mirbach, and Björn Ottersten. Kinect deform: enhanced 3d reconstruction of non-rigidly deforming objects. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 2, pages 7–13. IEEE, 2014.

[2] M Aggarwal and N Ahuja. Split aperture imaging for high dynamic range. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 10–17. IEEE, 2001.

[3] Manoj Aggarwal and Narendra Ahuja. Split aperture imaging for high dynamic range. *International Journal of Computer Vision*, 58(1):7–17, 2004.

[4] Kusuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. Photorealistic rendering for augmented reality using environment illumination. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 208–216. IEEE, 2003.

[5] Miika Aittala. Inverse lighting and photorealistic rendering for augmented reality. *The Visual Computer*, 26(6-8):669–678, 2010.

[6] Ibrahim Arief, Simon McCallum, and Jon Yngve Hardeberg. Realtime estimation of illumination direction for augmented reality on mobile devices. In *Color and Imaging Conference*, number 1, pages 111–116. Society for Imaging Science and Technology, 2012.

[7] M-O Berger. Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction. In *Computer Vision and Pattern Recogni-*

tion, 1997. Proceedings., 1997 IEEE Computer Society Conference on, pages 91–96. IEEE, 1997.

[8] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality: merging real and virtual worlds*. AK Peters/CRC Press, 2005.

[9] James F Blinn and Martin E Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[10] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam-learning a compact, optimisable representation for dense visual slam. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] Luca Bogoni. Extending dynamic range of monochrome and color images through fusion. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 7–12. IEEE, 2000.

[12] David E Breen, Ross T Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. In *Computer Graphics Forum*, volume 15, pages 11–22. Wiley Online Library, 1996.

[13] Francisc Bungiu, Michael Hemmer, John Hershberger, Kan Huang, and Alexander Kröller. Efficient computation of visibility polygons. *arXiv preprint arXiv:1403.3905*, 2014.

[14] Ricardo Cabral and Yasutaka Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635. IEEE, 2014.

[15] Dan A Calian, Kenny Mitchell, Derek Nowrouzezahrai, and Jan Kautz. The shading probe: fast appearance acquisition for mobile ar. In *SIGGRAPH Asia 2013 Technical Briefs*, page 20. ACM, 2013.

[16] Derek Chan, Hylke Buisman, Christian Theobalt, and Sebastian Thrun. A noise-aware filter for real-time depth upsampling. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*, 2008.

[17] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017.

[18] Chongyu Chen, Jianfei Cai, Jianmin Zheng, Tat-Jen Cham, and Guangming Shi. A color-guided, region-adaptive and depth-selective unified framework for kinect depth recovery. In *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, pages 007–012. IEEE, 2013.

[19] Li Chen, Hui Lin, and Shutao Li. Depth image enhancement for kinect using region growing and bilateral filter. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3070–3073. IEEE, 2012.

[20] Ji-Ho Cho, Satoshi Ikehata, Hyunjin Yoo, Margrit Gelautz, and Kiyoharu Aizawa. Depth map up-sampling using cost-volume filtering. In *IVMSP Workshop, 2013 IEEE 11th*, pages 1–4. IEEE, 2013.

[21] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2001.

[22] Ryan Crabb, Colin Tracey, Akshaya Puranik, and James Davis. Real-time foreground segmentation via range and color imaging. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–5. IEEE, 2008.

[23] Keenan Crane. "spot" 3d model. `https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/`[Accessed: 2018-10-24].

[24] Franklin C Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, 18(3):207–212, 1984.

[25] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas A Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, volume 2, page 10, 2017.

[26] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017.

[27] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.

[28] 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 annual conference on Computer graphics and interactive techniques*, pages 189–198. ACM, 1998.

[29] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 369–378. ACM Press/Addison-Wesley Publishing Co., 1997.

[30] Paul Joseph Diefenbach. *Pipeline rendering: interaction and realism through hardware-based multi-pass rendering*. PhD thesis, University of Pennsylvania, 1996.

[31] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):114, 2016.

[32] Chao Du, Yen-Lin Chen, Mao Ye, and Liu Ren. Edge snapping-based depth enhancement for dynamic occlusion handling in augmented reality. In *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, pages 54–62. IEEE, 2016.

[33] Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafał K Mantiuk, and Jonas Unger. Hdr image reconstruction from a single exposure using deep cnns. *ACM Transactions on Graphics (TOG)*, 36(6):178, 2017.

[34] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. In *ACM transactions on graphics (TOG)*, volume 23, pages 673–678. ACM, 2004.

[35] Stephen Ellis, Anthony Wolfram, and Bernard Adelstein. Three dimensional tracking in augmented environments: User performance trade-offs between system latency and update rate. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 46, 09 2002.

[36] Yuki Endo, Yoshihiro Kanamori, and Jun Mitani. Deep reverse tone mapping. *ACM Trans. Graph.*, 36(6):177–1, 2017.

[37] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 2018.

[38] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[39] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J. Brostow. Structured Completion of Unobserved Voxels from a Single Depth Image. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[40] Jan-Michael Frahm, Kevin Koeser, Daniel Grest, and Reinhard Koch. Markerless augmented reality with light source estimation for direct illumination.

In *Conference on Visual Media Production CVMP, London*, pages 211–220, 2005.

[41] Chunyu Gao, Yuxiang Lin, and Hong Hua. Occlusion capable optical see-through head-mounted display using freeform optics. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 281–282. IEEE, 2012.

[42] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017.

[43] S. Garrido-Jurado, R. Muaoz-Salinas, F.J. Madrid-Cuevas, and M.J. Maran-Jiminez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[44] Simon Gibson and Alan Murta. Interactive rendering with real-world illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 365–376. Springer-Verlag, 2000.

[45] Ned Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

[46] Thorsten Grosch, Tobias Eble, and Stefan Mueller. Consistent interactive augmentation of live camera images with correct near-field illumination. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 125–132. ACM, 2007.

[47] Jeffrey P Grossman and William J Dally. Point sample rendering. In *Rendering techniques*, pages 181–192. Springer, 1998.

[48] Lukas Gruber, Tobias Langlotz, Pradeep Sen, Tobias Höherer, and Dieter Schmalstieg. Efficient and robust radiance transfer for probeless photoreal-

istic augmented reality. In *2014 IEEE Virtual Reality (VR)*, pages 15–20. IEEE, 2014.

[49] Lukas Gruber, Thomas Richter-Trummer, and Dieter Schmalstieg. Real-time photometric registration from arbitrary geometry. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 119–128. IEEE, 2012.

[50] Lukas Gruber, Jonathan Ventura, and Dieter Schmalstieg. Image-space illumination for augmented reality in dynamic environments. In *2015 IEEE Virtual Reality (VR)*, pages 127–134. IEEE, 2015.

[51] Michael Haller, Stephan Drab, and Werner Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 56–65. ACM, 2003.

[52] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1397–1409, 2013.

[53] Anna Katharina Hebborn, Nils Höhner, and Stefan Müller. Occlusion matting: Realistic occlusion handling for augmented reality applications. In *Mixed and Augmented Reality (ISMAR), 2017 IEEE International Symposium on*, pages 62–71. IEEE, 2017.

[54] Gentaro Hirota, David T Chen, William F Garrett, Mark A Livingston, et al. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 429–438. ACM, 1996.

[55] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7312–7321, 2017.

[56] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2013.

[57] Xianyou Hou, Li-Yi Wei, Heung-Yeung Shum, and Baining Guo. Real-time multi-perspective rendering on graphics hardware. *Rendering Techniques*, 6:93–102, 2006.

[58] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 92–101. IEEE, 2016.

[59] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016.

[60] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[61] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile device. *IEEE Transactions on Visualization and Computer Graphics Proceedings International Symposium on Mixed and Augmented Reality 2015*, 22(11), 2015.

[62] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[63] Paul Kan and Hannes Kaufmann. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 133–141. IEEE, 2013.

[64] Peter Kán. Interactive hdr environment map capturing on mobile devices. In *Eurographics (Short Papers)*, pages 29–32, 2015.

[65] Peter Kán and Hannes Kaufmann. Physically-based depth of field in augmented reality. In *Eurographics (Short Papers)*, pages 89–92, 2012.

[66] Peter Kán, Johannes Unterguggenberger, and Hannes Kaufmann. High-quality consistent illumination in mobile augmented reality by radiance convolution on the gpu. In *Advances in Visual Computing*, pages 574–585. Springer, 2015.

[67] Masayuki Kanbara, Takashi Okuma, Haruo Takemura, and Naokazu Yokoya. A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In *Virtual Reality, 2000. Proceedings. IEEE*, pages 255–262. IEEE, 2000.

[68] Masayuki Kanbara and Naokazu Yokoya. Real-time estimation of light source environment for photorealistic augmented reality. In *ICPR (2)*, pages 911–914. Citeseer, 2004.

[69] Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High dynamic range video. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 319–325. ACM, 2003.

[70] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30, page 157. ACM, 2011.

[71] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic scene inference

for 3d object compositing. *ACM Transactions on Graphics (TOG)*, 33(3):32, 2014.

[72] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[73] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 85–94. IEEE, 1999.

[74] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 1–8. IEEE, 2013.

[75] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106. Citeseer, 2013.

[76] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Robust odometry estimation for rgb-d cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3748–3754. IEEE, 2013.

[77] Kiyoshi Kiyokawa, Mark Billinghurst, Bruce Campbell, and Eric Woods. An occlusion-capable optical see-through head mount display for supporting co-located collaboration. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 133. IEEE Computer Society, 2003.

[78] Georg Klein and Tom Drummond. Sensor fusion and occlusion refinement for tablet-based ar. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 38–47. IEEE, 2004.

[79] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[80] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[81] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems*, 2015.

[82] Jaroslav Křivánek and Mark Colbert. Real-time shading with filtered importance sampling. In *Computer Graphics Forum*, volume 27, pages 1147–1154. Wiley Online Library, 2008.

[83] Jean-François Lalonde. Deep learning for augmented reality. In *Proceedings of the 2018 Workshop on Information Optics*, 2018.

[84] Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. *arXiv preprint arXiv:1904.01175*, 2019.

[85] Vincent Lepetit and Marie-Odile Berger. A semi-automatic method for resolving occlusion in augmented reality. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 225–230. IEEE, 2000.

[86] Kenneth Levenberg. A method for the solution of certain non–linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, July 1944.

[87] David G Lowe. Object recognition from local scale-invariant features. In

Computer vision, 1999. The proceedings of the seventh IEEE international conference on, volume 2, pages 1150–1157. Ieee, 1999.

[88] Jiangbo Lu, Viet Anh Nguyen, Zeping Niu, Bhavdeep Singh, Zhiping Luo, and Minh N Do. Cutechat: a lightweight tele-immersive video chat system. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1309–1312. ACM, 2011.

[89] S Mann and R Picard. Being undigital with digital cameras. *MIT Media Lab Perceptual*, 1:2, 1994.

[90] Steve Mann, Corey Manders, and James Fung. Painting with looks: Photographic images from video using quantimetric processing. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 117–126. ACM, 2002.

[91] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[92] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In *SPBG*, pages 101–108, 2007.

[93] MATLAB. *MATLAB R2017a: Computer Vision Toolbox*. The MathWorks Inc., Natick, Massachusetts, 2017.

[94] Hiroto Matsuoka, Akirn Onozawa, and Eiichi Hosoya. Environment mapping for objects in the real world: a trial using artoolkit. In *Augmented Reality Toolkit, The First IEEE International Workshop*, pages 2–pp. IEEE, 2002.

[95] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J.Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? 2017.

[96] Maxime Meilland, Christian Barat, and Andrew Comport. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 143–152. IEEE, 2013.

[97] Abhimitra Meka, Maxim Maximov, Michael Zollhfer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[98] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[99] Paul Milgram, Haruo Takemura, Akira Utsumi, Fumio Kishino, et al. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–292, 1994.

[100] Gene S. Miller and C. Robert Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments: Siggraph course notes. ACM, 1984.

[101] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[102] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[103] Shree K Nayar and Tomoo Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 472–479. IEEE, 2000.

[104] Diego Nehab, André Maximo, Rodolfo S Lima, and Hugues Hoppe. Gpu-efficient recursive filtering and summed-area tables. *ACM Transactions on Graphics (TOG)*, 30(6):176, 2011.

[105] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

[106] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[107] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[108] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530. IEEE, 2012.

[109] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):169, 2013.

[110] Toshiya Okabe, Imari Sato, and Yoichi Sato. Spherical harmonics vs. haar wavelets: Basis for recovering illumination from cast shadows. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–50. IEEE, 2004.

[111] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the*

27th annual conference on Computer graphics and interactive techniques, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 2000.

[112] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[113] Vivek Pradeep, Christoph Rhemann, Shahram Izadi, Christopher Zach, Michael Bleyer, and Steven Bathiche. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 83–88. IEEE, 2013.

[114] Christoph Rhemann, Carsten Rother, Jue Wang, Margrit Gelautz, Pushmeet Kohli, and Pamela Rott. A perceptually motivated online benchmark for image matting. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1826–1833. IEEE, 2009.

[115] Grosch T Ritschel T, Dachsbacher C and Kautz J. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1), 2012.

[116] Kai Rohmer, Wolfgang Buschel, Raimund Dachselt, and Thorsten Grosch. Interactive near-field illumination for photorealistic augmented reality with varying materials on mobile devices. *Visualization and Computer Graphics, IEEE Transactions on*, 21(12):1349–1362, 2015.

[117] Kai Rohmer, Johannes Jendersie, and Thorsten Grosch. Natural environment illumination: Coherent interactive augmented reality for mobile and non-mobile devices. *IEEE transactions on visualization and computer graphics*, 23(11):2474–2484, 2017.

[118] Henry Roth and Marsette Vona. Moving volume kinectfusion. In *BMVC*, volume 20, pages 1–11, 2012.

[119] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation,

tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4471–4478, May 2017.

[120] Martin Rünz and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. *CoRR*, abs/1804.09194, 2018.

[121] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 157–164. IEEE, 2014.

[122] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[123] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *Visualization and Computer Graphics, IEEE Transactions on*, 5(1):1–12, 1999.

[124] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Stability issues in recovering illumination distribution from brightness in shadows. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–400. IEEE, 2001.

[125] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(3):290–300, 2003.

[126] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006.

[127] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 527–536. ACM, 2002.

[128] Alvy Ray Smith. Alpha and the history of digital compositing. `https://www.cs.princeton.edu/courses/archive/fall00/cs426/papers/smith95c.pdf`, 1995.

[129] Alvy Ray Smith and James F Blinn. Blue screen matting. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 259–268. ACM, 1996.

[130] Anthony Steed, Yonathan Widya Adipradana, and Sebastian Friston. The ar-rift 2 prototype. *2017 IEEE Virtual Reality (VR)*, pages 231–232, 2017.

[131] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 719–722. IEEE, 2011.

[132] William Steptoe, Simon Julier, and Anthony Steed. Presence and discernability in conventional and non-photorealistic immersive augmented reality. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 213–218. IEEE, 2014.

[133] Ivan E Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764. ACM, 1968.

[134] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[135] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[136] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.8.1 edition, 2016.

[137] Diego Thomas and Akihiro Sugimoto. A flexible scene representation for 3d reconstruction using an rgb-d camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2800–2807, 2013.

[138] Diego Thomas and Akihiro Sugimoto. A two-stage strategy for real-time dense 3d reconstruction of large-scale scenes. In *Workshop at the European Conference on Computer Vision*, pages 428–442. Springer, 2014.

[139] Diego Thomas and Akihiro Sugimoto. Parametric surface representation with bump image for dense 3d modeling using an rbg-d camera. *International Journal of Computer Vision*, pages 1–20, 2016.

[140] Michael D Tocci, Chris Kiser, Nora Tocci, and Pradeep Sen. A versatile hdr video production system. In *ACM Transactions on Graphics (TOG)*, volume 30, page 41. ACM, 2011.

[141] Jonas Unger and Stefan Gustavson. High-dynamic-range video for photometric measurement of illumination. In *Sensors, Cameras, and Systems for Scientific/Industrial Applications VIII*, volume 6501, page 65010E. International Society for Optics and Photonics, 2007.

[142] Jonathan Ventura and Tobias Höllerer. Online environment model estimation for augmented reality. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 103–106. IEEE, 2009.

[143] Hongcheng Wang, Ramesh Raskar, and Narendra Ahuja. High dynamic range video using split aperture camera. In *IEEE 6th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, Washington, DC, USA*. Citeseer, 2005.

[144] Jue Wang and Michael F Cohen. *Image and video matting: a survey*. Now Publishers Inc, 2008.

[145] Liang Wang, Minglun Gong, Chenxi Zhang, Ruigang Yang, Cha Zhang, and Yee-Hong Yang. Automatic real-time video matting using time-of-flight camera and multichannel poisson equations. *International journal of computer vision*, 97(1):104–121, 2012.

[146] Liang Wang, Chenxi Zhang, Ruigang Yang, and Cha Zhang. Tofcut: Towards robust real-time foreground extraction using a time-of-flight camera. In *Proc. of 3DPVT*, 2010.

[147] Oliver Wang, Jonathan Finger, Qingxiong Yang, James Davis, and Ruigang Yang. Automatic natural video matting with depth. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 469–472. IEEE, 2007.

[148] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.

[149] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: science and systems*, volume 11, 2015.

[150] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, page 0278364916669237, 2016.

[151] Matthias M Wloka and Brian G Anderson. Resolving occlusion in augmented reality. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 5–12. ACM, 1995.

[152] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.

[153] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér. Spatial-depth super resolution for range images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[154] Yiying Yao, Hidenori Kawamura, and Akira Kojima. Shading derivation from an unspecified object for augmented reality. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 57–60. IEEE, 2012.

[155] Xianghua Ying and Zhanyi Hu. Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model. In *European Conference on Computer Vision*, pages 442–455. Springer, 2004.

[156] Jingyi Yu, Jason Yang, and Leonard McMillan. Real-time reflection mapping with parallax. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 133–138. ACM, 2005.

[157] Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. Octree-based fusion for realtime 3d reconstruction. *Graphical Models*, 75(3):126–136, 2013.

[158] Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):174, 2016.

[159] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018.

[160] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and

Niloy J Mitra. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph.*, 31(4):99–1, 2012.

[161] Jiejie Zhu, Miao Liao, Ruigang Yang, and Zhigeng Pan. Joint depth and alpha matte optimization via fusion of stereo and time-of-flight sensor. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 453–460. IEEE, 2009.