



DOCTOR OF ENGINEERING (ENGD)

Graphics Insertions into Real Video for Market Research

Tarko, Joanna

Award date:
2020

Awarding institution:
University of Bath

[Link to publication](#)

Alternative formats

If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

GRAPHICS INSERTIONS INTO REAL VIDEO FOR MARKET RESEARCH

submitted by

JOANNA TARKO

for the degree of Doctor of Engineering

of the

University of Bath

Department of Computer Science

March 2020

COPYRIGHT

Attention is drawn to the fact that copyright of this thesis rests with the author. A copy of this thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with the author and that they must not copy it or use material from it except as permitted by law or with the consent of the author.

This thesis may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation with effect from (date)

Signed on behalf of the Faculty of Science

“To write a thing down is to control it
and sometimes to exorcise it.”

— Anthony Burgess
Murder to Music

ABSTRACT

Combining real videos with computer-generated content, either off-line (compositing) or in real-time (augmented and mixed reality, AR/MR), is an extensive field of research. It has numerous applications, including entertainment, medical imaging, education, sport, architecture, and marketing (advertising and commerce). However, even though well established in marketing as a part of a retail environment, there seem to be no known applications of merging real and virtual in market research.

The aim of market research is to help explain why a customer decided to buy a specific product. In a perfect scenario, study participants are placed in a real but fully controlled shopping environment, but in practice, such environments are very expensive or even impossible to build. Using virtual reality (VR) environments instead significantly reduces costs. VR is fully controllable and immersive but CG models often lack realism.

This research project aims at providing mixed-reality tools which combine real camera footage with computer-generated elements to create plausible but still controlled environments that can be used for market research. My work consists of the full graphics insertions pipeline for both perspective and 360° spherical cameras, with real-time user interaction with the inserted objects. It addresses the three main technical challenges: tracking the camera, estimating the illumination to light virtual objects plausibly, and rendering virtual objects and compositing them with the video in real-time.

Tracking and image-based lighting techniques for perspective cameras are well established both in research and industry. Therefore, I focused only on real-time compositing for perspective video. My pipeline takes camera tracking data and reconstructed points from external software and synchronises them with the video sequence in the Unity game engine. Virtual objects can be dynamically inserted, and users can interact with them. Differential rendering for image-based shadows adds to the realism of insertions.

Then I extend the pipeline to 360° spherical cameras with my implementation of omnidirectional structure from motion for camera tracking and scene reconstruction. Selected 360° video frames, after inverse tone mapping, act as spatially distributed environment maps for image-based lighting. Like in the perspective cameras case, differential rendering enables shadow casting, and user can interact with inserted objects.

The proposed pipeline enables compositing in the Unity game engine with correct synchronisation between the camera pose and the video, both for perspective and 360° videos, which is not available by default. This allows virtual objects to be inserted into moving videos, which extends the state of the art, which is limited to static videos only. In user studies, I evaluated the perceived quality of virtual objects insertions, and I compared their level of realism against purely virtual environments.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Dr Christian Richardt. He has an amazing eye for detail and is also infuriatingly right most of the time. My thesis, and every paper we wrote together, have benefited beyond imagination from his (sometimes brutally honest) comments. He will also always be my \LaTeX guru.

I thank my host companies, Dc-activ and Checkmate VR, and especially my industrial supervisors, Rob Thorpe and Tim Jarvis, for offering me this project and funding it. I am very grateful that they made me a part of their team and gave me unlimited opportunities to testing my research ideas.

I would also like to thank the Centre for Digital Entertainment and the EPSRC for funding the Engineering Doctorate program. I acknowledge the hard work of the CDE team: Sarah Parry, Becca Knight, Brent Kiernan and Zoe Leonard. They put a lot of effort into running the centre and organising cohort-building social events for the students to make us feel like one big family.

In the lab, I was never an island. Even if it sometimes felt like being a castaway, there were my fellow castaways around. I would like to thank you all for all the hugs, board games, talks in the kitchen and smiles (in no particular order): Maryam Naghizadeh, Anamaria Ciucanu, Bingjie (Bingo) Yu, Javier de la Dehesa Cueto-Felgueroso, Christina Keating, Luca Benedetti, Shahad Almansour, Latifa Al-shammari, Thomas Williams, John Benardis, Daniela De Angeli, Cillian Dudley, Zack Lyons, Dan Finnegan, Garoe Dorta Perez, Ben Pring, Sylwia Hyniewska, Catherine Taylor, Paul Sanderson, Paddy Boulton, Alexz Farrall, Thu Nguyen-Phuoc, Andreas Theodorou, Andrea Aller Tubella, Kenneth Cynric Dasalla, Murray Evans, Andrew Lawrence and Alessandro Di Martino.

Whenever I needed help in my research or advice, I could always count on the Computer Science Department staff: Neill Campbell (who led me through ups and downs of my first-year project with unflagging enthusiasm), Tom Haines, Martin Parsons, José Serra, Darren Cosker, Jo Hyde and Fabio Nemetz.

The list is so long because I was fortunate enough to get to know all these wonderful people who made my research journey bearable.

PUBLICATIONS

- **Dynamic Mixed-Reality Compositing with Unity**
Joanna Tarko, Christian Richardt, Peter Hall
Short paper and poster at the *14th European Conference on Visual Media Production (CVMP)*, December 2017
- **Environment Reflections with 360° Videos using Omnidirectional Structure from Motion**
Joanna Tarko, Christian Richardt
Short paper and poster at the *15th ACM SIGGRAPH European Conference on Visual Media Production (CVMP)*, December 2018
- **OmniMR: Omnidirectional Mixed Reality with Spatially-Varying Environment Reflections from Moving 360° Video Cameras.**
Joanna Tarko, James Tompkin, Christian Richardt
Short paper and poster at the *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, March 2019
- **Real-time Virtual Object Insertion for Moving 360° Videos**
Joanna Tarko, James Tompkin, Christian Richardt
In *Proceedings of the 17th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI)*, November 2019

CONTENTS

1	INTRODUCTION	1
1.1	Motivation	3
1.1.1	Market Research Tools for Conjoint Analysis	5
1.2	Research Objectives	10
1.3	Contributions	10
1.4	Dissertation Outline	12
2	BACKGROUND	13
2.1	Digital Compositing	14
2.2	The Fusion of Realities	15
2.2.1	Physical, Augmented, Mixed and Virtual Reality	15
2.2.2	The Beginnings of Augmented Reality	20
2.2.3	Augmented Reality in Marketing	22
2.3	Mixed Reality Rendering	25
2.4	Omnidirectional Content and Spherical Camera Model	28
2.4.1	Pre-digital Times	28
2.4.2	Spherical Camera Model	32
2.4.3	360° Image Representations	33
2.5	Omnidirectional Structure from Motion	39
2.5.1	Theoretical Foundations for Omnidirectional Structure from Motion	39
2.5.2	Image Correspondences	39
2.5.3	Feature Tracking	40
2.6	Real-time Photorealistic Graphic Insertions into 360° Videos	40
2.6.1	Illumination Estimation	40
2.6.2	Real-time 360° Virtual Object Insertion	41
3	DYNAMIC MIXED-REALITY COMPOSITING	43
3.1	Dynamic Mixed Reality Compositing Pipeline	44
3.1.1	Input Data Formats	45
3.1.2	Coordinate Systems	46
3.1.3	Camera Model	48
3.1.4	Synchronisation	48
3.1.5	Placing Objects in the Scene	52
3.1.6	Image-based Lighting	53
3.1.7	Differential Rendering	55
3.2	Experiments	58
3.2.1	Camera Tracked in Postproduction.	58

3.2.2	Camera Tracked on Set (experimental)	60
3.3	Discussion	61
4	USE CASE: PETROL STATION RESEARCH STUDY	63
4.1	User Study	64
4.1.1	Method	65
4.1.2	Analysis	70
4.1.3	Discussion	75
5	OMNIDIRECTIONAL STRUCTURE FROM MOTION	79
5.1	OmniSfM Pipeline	81
5.1.1	Feature Tracking	81
5.1.2	Epipolar Geometry	84
5.1.3	Multi-view Geometry	87
5.1.4	Bundle Adjustment	88
5.2	Experiments	89
5.2.1	Synthetic Data	89
5.2.2	Real Data	92
5.2.3	Comparison Between Hierarchical and One-Pass Bundle Adjustment	93
5.3	Discussion	95
6	REAL-TIME VIRTUAL OBJECT INSERTION FOR MOVING 360° VIDEOS	97
6.1	Virtual Object Insertion Pipeline	99
6.1.1	360° Video Stabilisation	100
6.1.2	Inverse Tone Mapping	100
6.1.3	Displaying 360° Videos in Unity	101
6.1.4	Image-based Lighting in Unity	103
6.1.5	Differential Rendering for Virtual Shadows	103
6.2	Experiments	105
6.3	Discussion	109
7	CONCLUSIONS	111
7.1	Inserting CG Objects into Real Footage	111
7.1.1	Standard Videos	112
7.1.2	360° Videos	113
7.2	A Real-Life Market Research Use Case	114
7.3	Limitations and Future Work	115
7.3.1	Research Challenges	115
7.3.2	Engineering Challenges	117
7.4	My Reflections on Conducting Research in Academia and in Industry	118
7.5	Concluding Remarks	119
	BIBLIOGRAPHY	120

Appendices	131
A	SCRIPTS FOR DYNAMIC COMPOSITING 133
A.1	Script for Exporting Camera Animation from 3ds Max to Unity 134
A.2	Synchronisation Unity Script for Standard Videos 136
A.3	Synchronisation Unity Script for 360° videos with Reflection Probes 139
B	APPENDICES FOR PETROL STATION RESEARCH STUDY 143
B.1	Participant information sheet and consent form 144
B.2	Qualifying questions 146
B.3	Demographic questions 147
B.4	First video 148
B.5	Perceptivity questions 149
B.6	Additional perceptivity question for mixed-reality video 151
B.7	First video played once more 152
B.8	Presence questionnaire 153
B.9	General preference questions 155

INTRODUCTION

With regard to visual effects, *compositing* is the process of combining visual content from different sources (live-action footage, still images and animations) so that they match each other in terms of colour, lighting, scale, perspective and camera movement. A vital part of compositing is to determine the parameters of the real-world camera used to record the footage in the process known as *matchmoving* (or *camera tracking*), and to transfer these to a virtual camera, used then to render the computer-generated (CG) elements. After that, a rendered image or animation is combined with the live-action footage using compositing software.

Compositing in visual effects usually means inserting CG objects in a photorealistic way and is an offline process. But in its general meaning, it refers to a rendering technique, sometimes also called *digital compositing*, which combines two or more images into one (see Section 2.1). Therefore, digital compositing is not confined to be offline only, and, in fact, is a vital part of augmented-reality and mixed-reality systems, which are real-time in their nature.

In augmented reality, computer-generated objects overlay, and add new information to the image of the real world (Figure 1.1). As there is much confusion about the naming convention and definitions of closely related augmented and mixed reality, I devote Section 2.2 to present my understanding of the topic, and the definitions I follow in this project.

Augmenting the reality does not necessarily mean inserting objects which are indistinguishable from the real ones. On the contrary, very often they are deliberately designed to attract attention, acting as hotspots in the image (Figures 1.1a to 1.1c). Photorealistic graphics insertions into real images, on the other hand, are designed in a way that allows the computer-generated elements to be indistinguishable from the real ones.

To insert virtual CG objects into real video footage in a photorealistic manner, we must solve three main technical challenges:

1. Tracking the video so that the virtual and real cameras share the same view,
2. Estimating the illumination to light virtual objects plausibly,
3. Rendering virtual objects and compositing them with the video in real time.

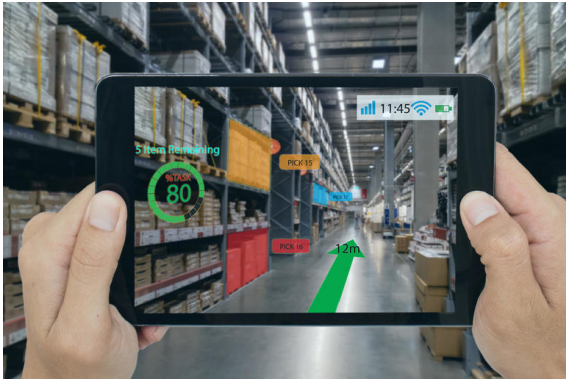
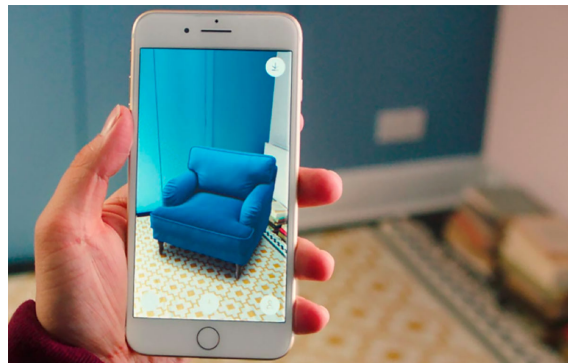
(a) A warehouse management system¹(b) A guide to the points of interest in the city²(c) An augmented reality game³(d) An interactive furniture shop assistant⁴

Figure 1.1: Examples of augmented reality applications.

The technique for a long time was reserved only for off-line applications, usually visual effects, due to low performance capabilities of mobile devices, which are the usual augmented reality platforms. Increasing computational powers of such devices seem promising for designing real-time photorealistic augmented reality systems, which have already drawn attention in the industry.⁵

¹ Image taken from <https://www.techrepublic.com/article/augmented-reality-for-business-cheat-sheet/>

² Image taken from <https://www.bazaarvoice.com/blog/ar-augmented-reality-shopping-retail/>

³ Image taken from <https://connected.messefrankfurt.com/en/2019/02/27/off-to-new-worlds-vr-and-ar-in-marketing/>

⁴ Image taken from <https://dmexco.com/stories/augmented-reality-in-marketing-8-current-examples-2/>

⁵ <https://www.foundry.com/trends/research/fame-project>, last accessed 05/09/2019.

1.1 MOTIVATION

In the standard approach to compositing, CG elements may be considered as “baked” in the image, because only rendered material from the virtual camera is used with fixed materials and textures, lighting, and positions of the CG elements. One may argue that an artist has full control over each of these elements and can adjust them according to the background live footage. However, any change in these parameters requires changing the virtual 3D scene and re-rendering it, which in case of complex scenes is very time-consuming. Usually, some manual adjustments to the final composite are also necessary, adding to the time spent on the task.



Figure 1.2: The components of compositing. CG elements are represented by their rendered 2D images and complementary mask (matte). Therefore, any change in these elements entails the necessity to update the rendering and the mask. *Image taken from Wright [117].*

Due to these limitations, in the film industry, where compositing is the most commonly used, there is a tendency to move as many postproduction elements as possible into the production or even preproduction stage of film making, as it significantly reduces the cost of changes made later in the pipeline. For example, prior knowledge of the final composition of a visual effects (VFX) shot allows the director, the director of photography, the VFX supervisor and even the actors to plan their work more efficiently. It also allows them to adjust it to avoid any unpredictable complications when it comes to the final compositing stage when any changes or improvements in the source material would be complicated and costly or even impossible to make. This approach is closely related to *virtual cinematography*, which is a term that describes any computer-based technique used in film production intended to replicate the real image or to create the image that would be impossible to shoot with the real camera [52]. Recently, the term *virtual production* was introduced to describe the whole process which combines postproduction and previsualisation techniques in the production stage of filmmaking. Virtual production utilises mixed reality rendering and real-time camera tracking to combine streaming from the film camera with CG elements, and sometimes with motion capture data. That allows

the film crew to see the visual effects previsualised on the film set as close as possible to how they look like after the postproduction process.

The issues described above can be encountered not only in the film industry but also wherever compositing finds its applications. One of them, which appeared recently, is modern market research.

The aim of market research is to design research to better understand the reason why customers purchase a specific item or choose a service and, therefore, to discover their buying preferences. The focus is on customers' actions, not their feelings about the product. A popular market research technique for this task is *conjoint analysis* [74]. It is based on stimuli, containing a set of attributes related to the product or service, and assessed by the shopper in order to make a decision. The aim is to indicate the most preferred combination of attributes, such as price, package design, or brand. In a perfect scenario, study participants would be placed in a real but fully controlled shopping environment, such as a supermarket with a shelf layout designed for the purpose. However, in practice, such environments are very expensive or even impossible to build.

Instead, different types of research surveys have been designed to implement conjoint analysis principles. With the advances in computer technology, eventually, most of the traditional surveys became computer-based, which makes them easier to conduct and analyse. The two market research companies, Checkmate VR⁶ and Dc-activ⁷, which are the host companies for my doctoral research, further exploited the opportunities given to them by 3D computer modelling and, recently, virtual reality (VR). They created innovative research tools by adapting virtual reality to conjoint analysis to enhance it with more degrees of realism and immersion.

Immersion is the ability of a virtual system to affect the senses, and therefore to engage the user with the virtual world as if it was real [7]. Closely related to immersion (and often confused with it) is *presence*, which means "one's sense of being in the virtual world" [7]. Different levels of immersion can be linked to corresponding levels of presence [89]. A component of presence, called "place illusion", can be explained as people's belief of being somewhere else [88]. Place illusion, together with plausibility illusion (i.e. plausible events taking place in a virtual world) leads to users behaving naturally, even if they know their surroundings are artificial [88, 89].

Using computer-generated environments in market research is based on the assumption that, if the participants experience a sufficiently high level of presence, and are presented a plausible study scenario, they will behave in the same way as they would in the real world. Virtual shopping environments created by Checkmate VR and Dc-activ are modelled at different levels of accuracy, depending on the study purpose.

Currently, levels of conjoint analysis may vary from a set of photographs of products to a fully shoppable virtual store environment if VR techniques are implemented. They may

⁶ <http://www.checkmatevr.com>

⁷ <http://dc-activ.com>

represent new products, different shelf display configurations, whole shop layouts or even new building designs such as petrol stations.

1.1.1.1 *Market Research Tools for Conjoint Analysis*

Below is a short overview of types of market research tools, including traditional and computer-based methods, and among them those which use virtual reality:

1.1.1.1.1 *Paper-based surveys (obsolete)*

In paper-based surveys, study participants are given print-outs with pictures of the products to choose the one they would buy. This type of survey is now obsolete as most modern market research tools (and all listed below) are computer-based.

1.1.1.1.2 *A/B testing*

In this type of survey, the participants are shown pairs of images, and they have to choose one image of each pair they like more. The goal of the survey is to directly compare two products or versions of a product against each other.

1.1.1.1.3 *Multivariate regression*

In a multivariate regression based survey, there are multiple versions of one product. The aim of the survey is to find the optimal combination of product attributes. An example is shown in Figure 1.3. The study can be conducted on a computer screen or on a virtual reality headset for complete immersion.

1.1.1.1.4 *Four and none (4/o) conjoint*

This survey gives the participants a choice of four different products, while they can select one or none of them. Figure 1.4 shows an example of 4/o conjoint with four frozen food products.

1.1.1.1.5 *Virtual shelf conjoint*

Virtual shelf conjoint is a more advanced tool to measure consumers' buying preferences. It requires creating a 3D model of the whole shelf or the aisle in the shop as a shopping context. Products are then placed on shelves in different configurations (sometimes next to the competitive products for comparison). The study participant can pick a virtual product and rotate it for a better feeling of realism before deciding if to place it in the shopping basket or to put it back on the shelf (Figure 1.5).

Similarly as in the case of multivariate regression survey, the study can be conducted either on a computer or on a VR headset.

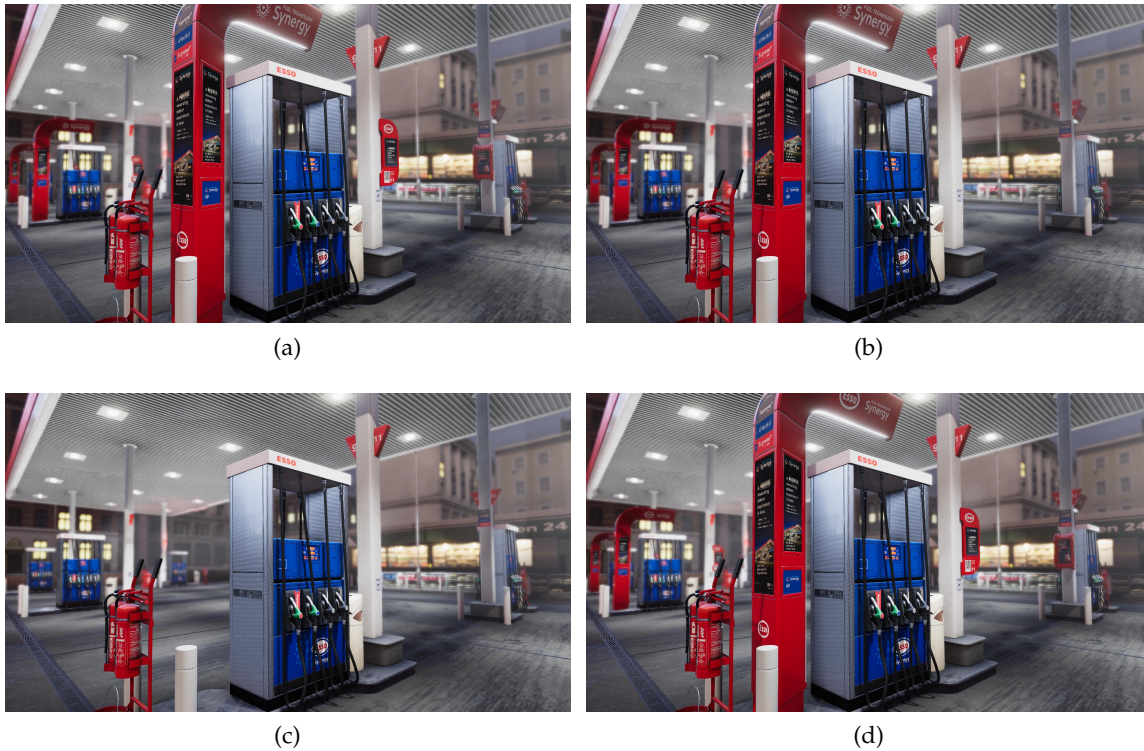


Figure 1.3: Assets for a multivariate regression survey on petrol station revamping: (a) a petrol pump with a 'koala' advertising panel mounted on the canopy columns, and a 'wave' - an inverted L-shape alongside and over the pumps, (b) the same without 'koala', (c) the same without 'koala' and 'wave', (d) the same but with a different version of the logo on the 'wave'. Image courtesy of Checkmate VR and Dc-activ.

1.1.1.6 Virtual shelf combined with 4/o choice

This type of survey extends the virtual shelf with 4/o choice. After picking a product from the shelf, the study participant is given three more options to choose from.

1.1.1.7 Customer journey

In some studies, elements of conjoint analysis may also include the customer journey, which is the whole way from the customer's house to the store. It contains both indoor and outdoor environments, and all kinds of stimuli that the customer may encounter in the real world (such as posters, advertisements on buses and bus stops, radio broadcasts, ads in their mobile phones and much more).

Virtual reality environments are fully controllable and immersive, and therefore are a good choice for modelling real environments for user studies. However, while virtual

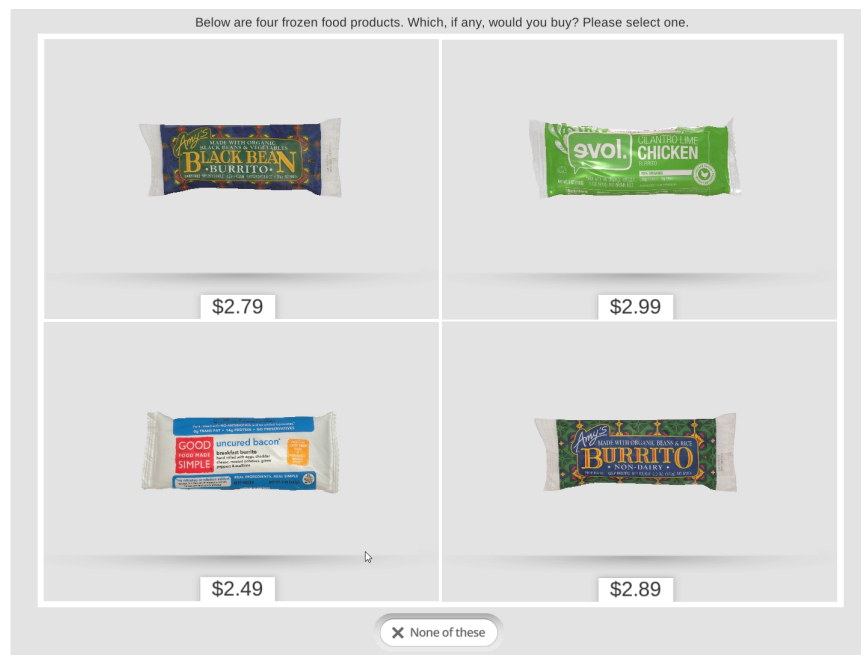


Figure 1.4: Four and none (4/o) choice. The study participant can choose one of the four options or none of them. *Image courtesy of Checkmate VR and Dc-activ.*

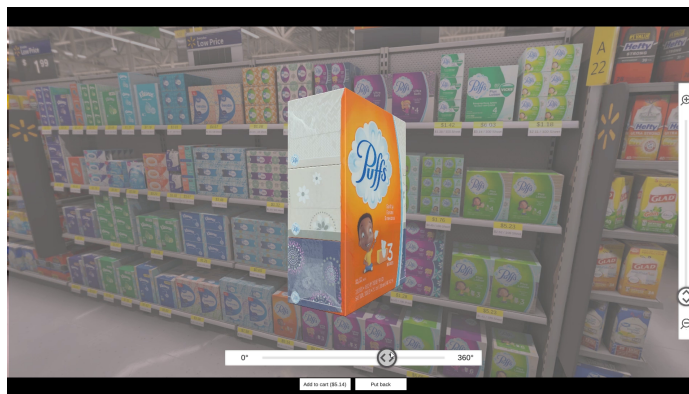
indoor environments proved to be close enough in perception from the real environments⁸, outdoor scenes are more challenging with their different lighting models, foliage elements, and a scale size they should be modelled with. Apart from presence, visual realism is another important factor to ensure the realistic behaviour in immersive virtual environments [91]. Thus, only capturing as much of the surroundings as possible would provide the study participant with the immersive experience of being in a real place. These factors combined make the outdoor virtual reality environment less plausible for the participants, and may interfere with the research results. Using real-life footage instead of a purely virtual scene would increase the degree of realism of the test, while reducing the complexity and size of the project at the same time.

Stimuli in the shape of posters or billboards are an integral part of the customer journey, but in the case of the real footage, it is cumbersome to place the real posters in the recorded scene. They should also change depending on the scenario assigned to a particular study participant, which usually requires a large number of combinations (see multivariate regression and conjoint types of market research surveys above). While it is not a problem in CG environments, which can dynamically change according to the pre-programmed design assigned to the study scenario, it is practically impossible to do in the real world. On the other hand, the greater realism of the real environments compensates for these

⁸ Dc-activ's internal research.



(a)



(b)

Figure 1.5: Virtual shelf conjoint: (a) supermarket shelves packed with products (one of the possible configurations), (b) a product picked by the study participant, who can interact with it (rotate it and zoom in/out to it). *Image courtesy of Checkmate VR and Dc-activ.*

shortcomings and motivates to search for a solution to overcome them. It must also be taken into account that conjoint analysis scenarios are usually completed online, and therefore the project size and the overall loading time is of extreme importance. A new approach to this problem should address these requirements and make it possible to control a wide variety of market research scenarios.

If recording multiple versions of the real scene is not an option, we may borrow from the experience of the filmmaking industry, and create the desired scenes by compositing virtual elements onto real-life footage. Compositing tools for market research would not be then much different from their applications in entertainment, save for the need for many different versions of the same scene. As I mentioned at the beginning of this section, classic compositing is very inefficient and time-consuming in this case, because every version requires an equal amount of effort to produce, but most of the content of the scene does

not change between versions. However, if classic compositing could be combined with elements of virtual production, and especially real-time insertions using augmented and mixed reality rendering, this waste of time and resources can be minimised.

Using augmented reality and associated compositing tools in marketing is not a new idea. Nonetheless, in spite of their increasing popularity, they are used mostly for advertising, engaging customers in the product or the brand campaign or providing them with a previsualisation of the products (see Section 2.2.3). Augmented reality is, therefore, a part of the retail environment and can influence the customers' choices, but it seems that no use of AR in market research has ever been explored.

The recent development of image capturing technologies offers new opportunities for immersive content creation. 360° spherical cameras, which are cameras that capture everything around them in every direction (see Section 2.4), are rapidly gaining in popularity and may provide a suitable replacement for VR content in terms of immersion. They also require revising the existing compositing and camera tracking algorithms and updating them to fit the different camera model. That opens up whole new directions of computer vision research.

In Table 1.1, I compare the strengths and weaknesses of fully computer-generated environments vs. mixed-reality environments that should be taken into account when building a market research environment. The comparison turns out to be in favour of mixed-reality environments.

Table 1.1: The comparison of strengths and weaknesses of using fully computer-generated (CG) and mixed-reality (MR) environments in building video assets for market research study.

	CG	MR
Low cost of building	✗	✓
Changes made in real time (Flexibility)	✓	✓
Realism	✗	✓
Immersion	✓ (when viewed in a VR headset)	✓ (with 360° video, when viewed in a VR headset)
Interaction with the environment	✓	✓ (only with inserted elements)
Image quality	✓	✓ / ✗ (depends on the camera, still relatively low for 360°)

1.2 RESEARCH OBJECTIVES

This research project is carried out in collaboration with two market research companies, Checkmate VR and Dc-activ. After a series of talks between the companies representatives, myself and my academic supervisors, we formulated the prime objective of this project:

To make computer graphics inserts into real video footage, to create a plausible result. By plausible, it means that at the very least, the insertions should look sufficiently realistic so that meaningful marketing experiments can be conducted using the edited video.

The computer graphics objects can range from texture replacement (as on the screen of a mobile phone) through to complete three-dimensional models of a building (such as a petrol station). More commonly, billboards, posters, and individual product items comprise the graphics models. More demanding is a set of insertions with which a participant can interact.

In addition, the project should be integrated with the companies' content production pipeline, and especially it should be compatible with the Unity game engine used by them to create research environments.

After the first stage of the project, where standard perspective videos were considered, the project specifications were extended with the potential use of 360° spherical videos as a replacement for fully computer-generated virtual reality environments.

The research objectives of this project are then as follows:

- O1: To design a method to insert CG objects into real footage that works for both standard and spherical videos. The design process requires addressing the technical challenges related to camera tracking, lighting estimation and compositing in real time. To fulfil the plausibility requirement, all the insertions must be the photorealistic ones.
- O2: To test the method on a real world market research use case.
- O3: To compare the existing virtual reality market research environments created by my industrial collaborators with the new mixed-reality ones created using my method. In particular, to test if the research study participants behave more naturally in mixed-reality environments than they do in purely CG ones or, at least if they behave in the same way in both types of environments.

1.3 CONTRIBUTIONS

In this dissertation, I explore tools for combining camera footage with computer-generated elements to create realistic but still controllable environments that may be used for market research. They advance the state-of-the-art fully computer-generated environments for market research surveys with an additional degree of realism. The study comprises standard perspective camera models as well as 360° spherical cameras.

To enable dynamic mixed-reality compositing for standard video in real time, I present the following contributions (published as CVMP short paper [100]):

- A mechanism to reliably synchronise event-driven animation with clock-driven live-action video footage (Chapter 3),
- A complete pipeline for real-time dynamic mixed-reality compositing in a state-of-the-art game engine (Unity), including differential rendering for image-based shadowing (Chapter 3).

The current real-time dynamic compositing systems, used for example in visual effects previsualisation on set, work with real-time streaming from the camera. To the best of my knowledge, this is the first time a game engine was used with a pre-tracked camera for real-time compositing.

I also extend this pipeline to 360° spherical cameras. I propose solutions to all three main technical challenges associated with inserting CG objects into real footage, in the context of 360° videos. These are: tracking of a moving 360° video camera, reconstruction of HDR spatially-varying environment maps for image-based lighting, and real-time rendering and compositing of virtual elements and real footage using the Unity game engine (early stages published as CVMP short paper [99], and IEEE VR short paper [101], the full pipeline published as VRCAI paper [102]). I contribute:

- An offline structure-from-motion pipeline for 360° videos, which tracks directly on stitched equirectangular video. Unlike most other pipelines (academic and commercial), my approach does not require unstitched images or remapping the equirectangular images to the cubemap format as input, which reduces the number of steps. My pipeline recovers the camera motion and reconstructs the sparse structure of the environment (Chapter 5). This enables me to match the view of virtual CG objects to a 360° video of a moving camera. It can also be applied to generically stabilise 360° video. I designed the pipeline by combining steps from other, sometimes partial, pipelines from literature. This simplified the algorithm and made it easier to implement.
- An offline recovery of spatially-varying high dynamic range environment maps via inverse tone mapping, which plausibly reproduces real-world lighting along the camera path (Chapter 6). State-of-the-art systems [69, 80] use one global environment map, which produces less accurate reflections than a set of spatially-varying maps.
- Real-time rendering of mixed-reality objects into omnidirectional moving video with dynamic image-based lighting and shadowing, built into the interactive Unity game engine (Chapter 6). It extends the current state-of-the-art system for static cameras, MR360 [80], to moving cameras.

These contributions expand the use and flexibility of 360° video for interactive computer graphics and visual effects applications.

1.4 DISSERTATION OUTLINE

This dissertation is structured chronologically to demonstrate how new modules added to the project built on the basis of the previous work as the project advanced. Thus, the remainder of this dissertation is organised as follows:

Chapter 2 contains the theoretical foundations and the literature review relevant to the parts of the project described in later chapters.

In **Chapter 3**, I describe the dynamic compositing pipeline for standard videos, which forms the foundation for the extended version for omnidirectional videos described in **Chapter 6**. **Chapter 4** contains a real-world use case for the pipeline, together with a user study designed to compare it with a similar study conducted by my host company using purely computer-generated environment.

Chapter 5 contains an implementation of omnidirectional structure from motion, and together with **Chapter 6** forms a description of a complete pipeline for inserting virtual objects into moving 360° videos.

Chapter 7 concludes the dissertation and provides directions for future work. It also includes my reflections on different styles of conducting research in industry and in academia, and the position of an EngD researcher in-between.

BACKGROUND

In this chapter, I collected background information and related work relevant to the content described in the later chapters of this dissertation. The chapter is structured as follows:

Section 2.1: *Digital Compositing*

This section contains the basics of the technique of blending two or more images, which is essential to image overlays.

Section 2.2: *The Fusion of Realities*

In this section, I provide a general introduction to the topic of mixing real and computer-generated imagery. It also includes definitions of closely related augmented, mixed and virtual reality, followed by a brief overview of the beginnings of augmented reality. The final part of this section focuses on applications of augmented reality in marketing and commerce, which are the most relevant to the topic of my project.

Section 2.3: *Mixed Reality Rendering*

This section focuses on a specific type of graphics insertions into real images, which makes them indistinguishable from reality. Its theoretical background is based on Debevec's concept of the light-based scene model.

Section 2.4: *Omnidirectional Content and Spherical Camera Model*

This section defines omnidirectional cameras and introduces the spherical camera model as their subclass. It also discusses different spherical image representations and mappings between them and mentions their relevance to specific parts of my project.

Section 2.5: *Omnidirectional Structure from Motion*

In this section, I collected work that provides theoretical foundations for omnidirectional structure from motion. I also discuss methods for finding image correspondences and feature tracking in the context of spherical videos.

Section 2.6: *Real-time Photorealistic Graphics Insertions into 360° Videos*

This section presents state-of-the-art approaches to real-time and interactive

graphics insertions into spherical videos. It also discusses methods for estimating the illumination of the environment for the highest possible visual fidelity of inserted objects.

2.1 DIGITAL COMPOSITING

The term compositing used in the context of visual effects production is much broader than the one related to rendering techniques. The latter is blending two or more images, which can be considered as the last step of compositing visual effects. Sometimes the word “digital” is added to distinguish computer-based compositing from previous image blending techniques performed with the use of optical printers and film stock.

The list of milestone papers in the history of digital compositing was neatly compiled by Mark Levoy on his project website¹. He acknowledged main contributors to the technique, who shaped it as we know it today.

Linear interpolation (lerp) of two images was first mentioned by Smith [92], who gave the formula, which is now known as *matting* (or *compositing*) *equation* (see illustration in Figure 2.1):

$$\mathbf{I} = \alpha \cdot \mathbf{F} + (1 - \alpha) \cdot \mathbf{B}, \quad (2.1)$$

where \mathbf{I} is the final image, $\alpha \in [0, 1]$ is an alpha matte (opacity) of the foreground, and \mathbf{F} and \mathbf{B} are the foreground and background images, respectively. It can be rewritten in its per-pixel form:

$$I(x, y) = \alpha(x, y) \cdot F(x, y) + (1 - \alpha(x, y)) \cdot B(x, y), \quad (2.2)$$

where (x, y) are the image coordinates of a pixel, usually omitted for clarity.

This was the first time when alpha α (or A) was considered as a part of the image, the fourth channel after R (red), G (green) and B (blue).

Wallace [111] extended the equation above with a second alpha matte for the background, considering both foreground and background as partially opaque:

$$\alpha = 1 - (1 - \alpha_f) \cdot (1 - \alpha_b) \quad (2.3)$$

$$\mathbf{I} = (\alpha_f \cdot \mathbf{F} + (1 - \alpha_f) \cdot \mathbf{B} \cdot \alpha_b) / \alpha \quad (2.4)$$

where α_f and α_b denote the foreground and the background alpha, respectively.

Finally, Porter and Duff [78] simplified the equation by introducing pre-multiplication of the foreground and the background by their alpha channels:

$$\alpha = \alpha_f + (1 - \alpha_f) \cdot \alpha_b \quad (2.5)$$

¹ <https://graphics.stanford.edu/papers/merging-sig81/>

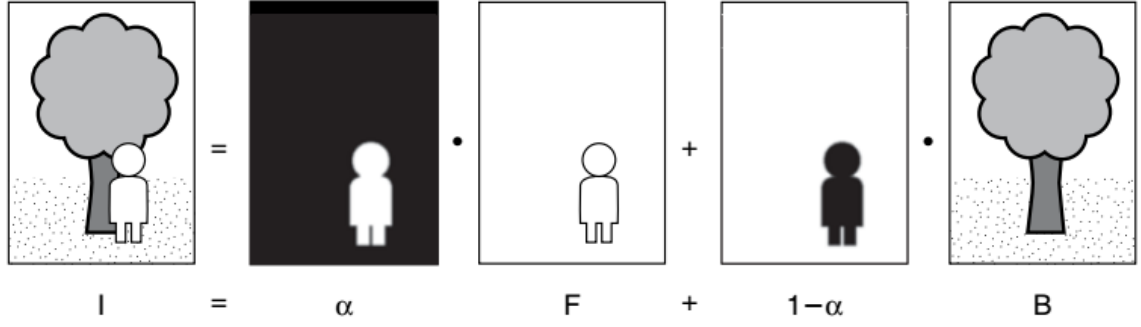


Figure 2.1: Illustration of the compositing equation. The final image I is formed by adding a foreground F , multiplied by the alpha matte α , and a background B , multiplied by the inverted alpha $(1 - \alpha)$. Figure taken from Radke [79].

$$I' = F' + (1 - \alpha_f) \cdot B' \quad (2.6)$$

where $F' = \alpha_f \cdot F$, $B' = \alpha_b \cdot B$ and $I' = \alpha \cdot I$.

Duff [22] then extended this work to compositing 3D rendered images using the z-buffer for anti-aliasing, and representing the image using five channels, including depth, $RGB\alpha Z$ (RGBAZ).

2.2 THE FUSION OF REALITIES

Whenever a computer-generated object is inserted into a real image in real time, with an awareness of physical space and its contents, we observe the two types of realities interlacing. The first one is *physical* reality, which is the world how we directly perceive it with our senses or captured by recording devices. The second one is *virtual* reality (VR), meaning the world generated entirely by computer. The fusion of these two results in creating new types of “realities”.

2.2.1 Physical, Augmented, Mixed and Virtual Reality

Between the physical reality, and virtual reality, there is a spectrum of intermediate states, called mixed reality, where these two realities overlap (Figure 2.2). These states belong to either augmented reality (AR) or augmented virtuality (AV), depending on how close they are to each end of the spectrum, called Reality-Virtuality Continuum [70, 71].

Whenever a computer-generated element overlays an image of the real world, and carries some additional information or complements or extends the image, the term “augmented reality” is used to describe the final effect. Augmented virtuality, on the other hand, is

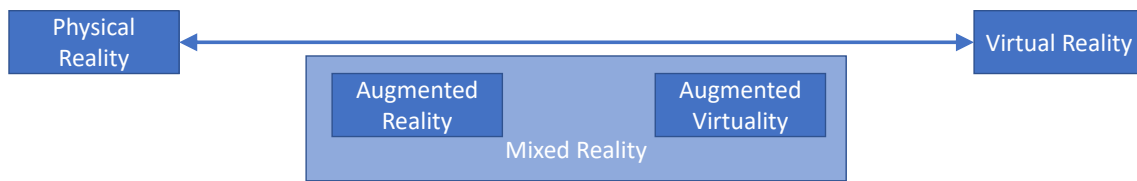


Figure 2.2: The Reality-Virtuality Continuum defined by Milgram and Kishimo [70].

placing real-world elements inside VR environments (an example is a virtual studio set, as commonly used in TV shows). The term is less popular than AR, and very often mixed reality is used instead to describe any system where the two realities, physical and virtual, interlace.

Mann [65], in turn, introduced the more general term *mediated reality*, which encapsulates virtual, augmented and mixed reality (Figure 2.3), but also modulated reality. One example of the latter is *diminished reality*, where objects are removed from an image rather than inserted into it. This classification model emphasises the ability of mediated reality to change or filter the physical reality by an intermediary device.

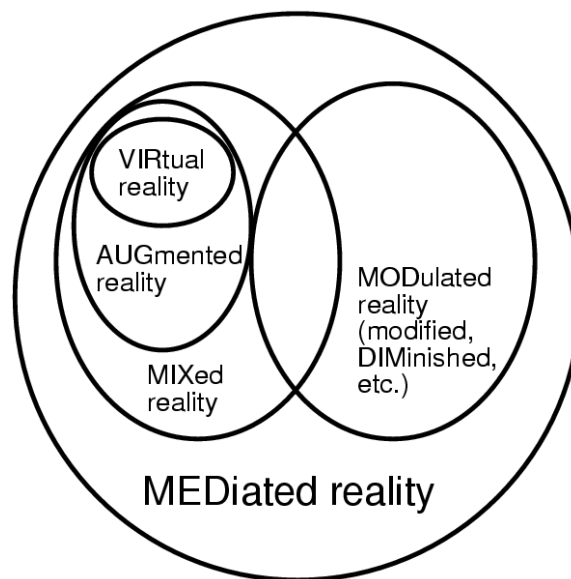


Figure 2.3: According to Mann, mediated reality encapsulates virtual, augmented and mixed reality, but also modulated reality, which contains other types of changing the reality that do not fall into the mixed reality category [65].

The intermediary devices that are able to alter the physical reality can be divided into the following categories:

- Home computers

- Mobile devices (smartphones, tablets)
- Wearables (e.g. head-mounted displays - HMD, AR glasses)
- Consoles

Milgram and Kishimo's Reality-Virtuality Continuum proved to be more popular than Mann's mediated reality, and since its publication in 1994, it provided the most commonly used definition for mixed-reality systems. However, nowadays, after twenty-five years of hardware and software development, a question arises: does mixed reality still mean the same as in the past?

2.2.1.1 *What Is Mixed Reality?*

Mixed reality is often defined as a 'live' technology, in which all processing, including video streaming, is done in real time. In other cases, real-time interaction between the elements of different realities, including the user, is sufficient to call the system "mixed reality". This ambiguity results from the fact that in the past twenty-five years, the technology has evolved, but the official definitions have not. There are several working definitions of mixed reality and their different interpretations, but not a single one to cover the entire vast field. Therefore, it sometimes becomes difficult to classify a system unambiguously as mixed reality. A good example of the ways in which the term can be interpreted or even exploited is the case of Microsoft and their mixed-reality headsets.

In 2015 Microsoft announced the HoloLens, a combination of a head-mounted display and AR glasses, which was the first headset advertised as a mixed-reality device². That has popularised the term alongside the already well-established augmented and virtual reality. Mixed reality started to be associated with the evolution of augmented reality, and with systems which are able to produce a new, better quality of augmented imagery.

However, two years later, Microsoft introduced another hardware, called Windows Mixed Reality immersive headset, and that name caused a lot of confusion. The headset, even though it used the same Windows Holographic platform as the HoloLens, was capable of handling only virtual reality. Therefore, calling it "mixed reality" might be considered misleading.

The explanation lies in the interpretation of the definition of mixed reality. Microsoft defines it as a combination of human, computer and environment factors interacting with each other in one system (see Figure 2.4). They also claim that both their devices lie on Milgram and Kishimo's Reality-Virtuality continuum, where, in case of the second headset, the physical world was replaced with a sense of "presence" of a virtual world [9]. Nevertheless, they presented a slightly modified continuum model. Virtual reality was moved to the spectrum as a range of states rather than a single point, and replaced by *digital world* as an endpoint of the continuum (see Figure 2.5).

² <https://www.theverge.com/2015/1/21/7867593/microsoft-announces-windows-holographic>, last accessed 12/09/2019 (text and images, missing video content.)

This example drew attention to the lack of a precise definition of the term mixed reality and to the associated room for interpretations, which in case of Microsoft were justified solely by marketing purposes. The company was not entirely to be blamed, as in a field that is constantly changing, old definitions do not always keep up with the speed of its evolution.

A recent paper by Speicher et al. [94] aims at updating the terminology established by Milgram and Kishimo [70], especially in terms of mixed reality and how the term is interpreted in academia and industry nowadays. By interviewing a group of experts in the field, the authors found that there is a need for a single definition in the academic and the industry community. However, they concluded that this task is unlikely to be accomplished. This is because all interested parties understand “mixed reality” in a slightly different way.

By analysing the papers accepted to four top conferences in the field of mixed reality (CHI, CHI PLAY, UIST and ISMAR), and the outcome of the aforementioned interviews, Speicher et al. indicated that the modern definition of mixed reality varies depending on the application and the context. They identified six working definitions:

1. MR as a part of Reality-Virtuality continuum defined by Milgram [70, 71].
2. MR as a synonym for AR.
3. MR as a collaboration between physically remote users through linking AR and VR environments.
4. MR as a combination of AR and VR existing in one system or device, not necessarily at the same time.
5. MR as an alignment of a physical and a virtual environment.
6. MR as an evolved version of AR with better spatial understanding and interaction between the real and the virtual objects and the user.

In the context of my work, I base my definition of mixed reality on a combination of some of these mentioned above. In particular:

- I acknowledge the Reality-Virtuality continuum as a blend of the real and the virtual content.
- I associate mixed reality with an extended version of augmented reality, where objects are inserted into real footage in a photorealistic way, with spatial understanding of the scene, and with user interaction.
- I consider immersion as a “weak” version of physical reality. Therefore, I agree with Microsoft’s argumentation that the feeling of presence can replace the real world. It can be achieved with 360° videos, even if they are not streamed in real time.

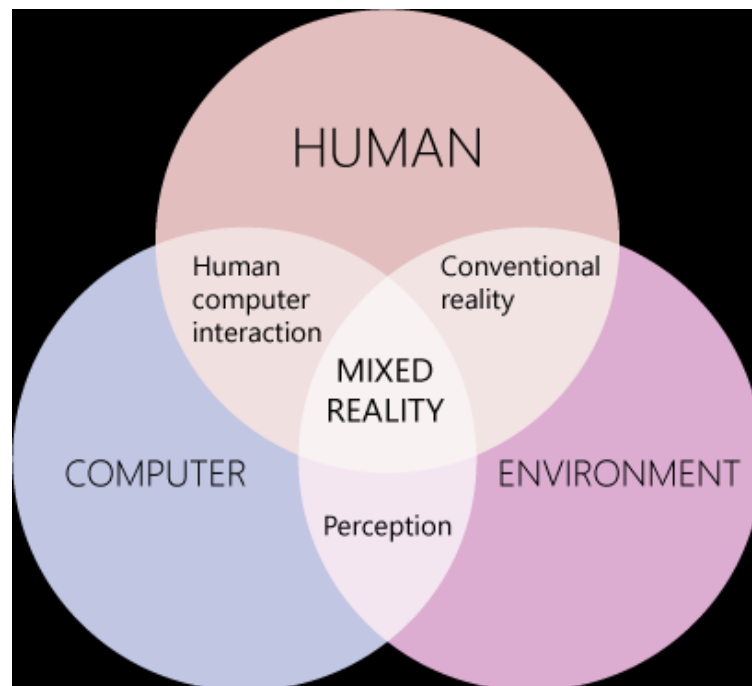


Figure 2.4: Mixed reality as a combination of three factors: human, computer and environment, and their interaction in one system. *Figure taken from Bray and Zeller [9].*

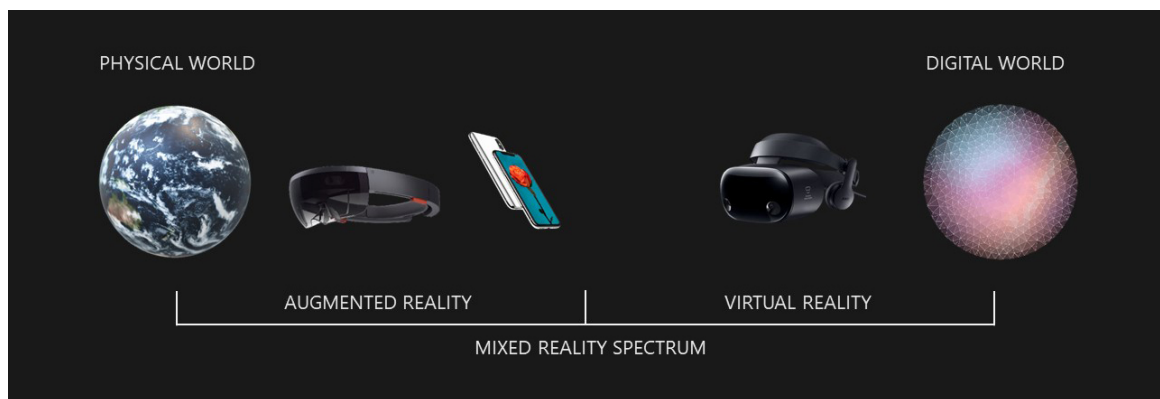


Figure 2.5: Where Microsoft's devices exist on the mixed-reality spectrum. Augmented reality and virtual reality are both depicted as a subset of the spectrum. The HoloLens headset and mobile devices occupy the augmented reality end of the spectrum, while Windows Mixed Reality headset lies on the virtual reality part. *Figure taken from Bray and Zeller [9].*

- In relation to the previous point, I define a mixed reality system as a combination of mixed reality rendering and 360° videos, as proposed by Rhee et al. [80].

In my opinion, the real-time interaction with inserted objects and between the real and the virtual elements is more important than real-time streaming of the real content to call a system mixed reality.

2.2.2 *The Beginnings of Augmented Reality*

For a long time, before mixed reality became popular, augmented reality was the only term to describe inserting computer-generated content into real video in real time. Summarising its history is a challenging task. The first difficulty arises when trying to determine the moment in time when AR appeared for the first time. The earliest well-known example of seeing things where they are not is so-called “Pepper’s ghost”, an optical illusion known since the 16th century and popularised three hundred years later by John Henry Pepper in theatres. It involves a clear glass or similar transparent item placed at 45° between the viewer and an empty scene, producing faint reflections of objects outside the viewer’s field of view, so they appear as “ghosts” populating the scene.



Figure 2.6: An illustration of Pepper’s ghost effect used on stage³. The faint reflection of a hidden actor on a glass screen appears to the audience as a ghost.

³ <http://scihi.org/john-henry-pepper-peppers-ghost/>, last accessed 03/09/2019



Figure 2.7: An American football TV broadcast from 1988 was the first AR application available for a wider audience⁴.



Figure 2.8: Ivan Sutherland's "The Sword of Damocles", the first head mounted display in history [98]. ©Photo by Ivan Sutherland

Nevertheless, even if we narrow down the definition of AR only to computer-generated content which overlays the physical reality, as described in the previous section, opinions differ. Sometimes an American football game TV broadcast from September 27, 1988, is believed to be the first example of augmented reality in history. On the screen, a yellow line was drawn on the field to help the television viewers to notice where the players should be to score points (Figure 2.7). Unquestionably, it was the first AR application available for a wider audience in their everyday life. However, despite this fact, the first concepts of augmented and virtual reality appeared as far back as in 1965 [97], followed by the construction of the first head-mounted AR/MR display by Ivan Sutherland in 1968 [98] (Figure 2.8).

As it is common for emerging technologies, AR almost immediately found its way into military applications, such as research on aircraft cockpit integrated with HMD, which allowed the computer graphics to overlay the pilot's view and to help in managing the amount of visual data given to them [32, 33]. Simultaneously, a contrasting research was conducted on using computer-generated insertions into real video for interactive art [56].

However, it is true that the term "augmented reality" was coined no sooner than in 1992 [12], when AR started to appear in industrial applications due to the increasing capabilities of computers. Since then, the popularity of AR rapidly increased and the various applications of the technology now include medical imaging, education, entertainment, advertising and commerce, tourism, sport, architecture and design, and navigation.

2.2.3 *Augmented Reality in Marketing*

Amongst the many applications of AR, which appear in various papers, books or surveys, marketing has an important place after entertainment, sport, medicine and education. In their survey on augmented reality, Billingham et al. [8] dedicate an entire section to its use in marketing. Similarly, Schmalstieg and Höllerer [85] mention advertising and commerce on their list of AR applications.

Both publications agree on AR being a tool which enables the interaction of customers with a product and therefore encourages them to learn more about it and spread the news by word of mouth. Augmented print, such as books, magazines, posters or product packages, is the most commonly mentioned form of marketing application of AR. It can move still images⁴, project virtual 3D objects directly onto printed surfaces, where they can sometimes be customised⁵, or transfer customers into the virtual world where they can play games based on the merchandise. It can even display short 3D animations to build positive brand associations⁷. Besides, AR can help with the buyer's decision process,

⁴ <http://entertainment.howstuffworks.com/first-down-line.htm>, last accessed 05/09/2019.

⁵ <https://www.youtube.com/watch?v=-JeygBEcNDE>, last accessed 17/03/2020.

⁶ <https://vimeo.com/4233057>, last accessed 05/09/2019.

⁷ <https://www.mobilemarketer.com/news/coca-cola-cans-activate-animated-stories-in-ar/562655/>, last accessed 01/10/2019.

being present at trade shows and showrooms, where it reduces the cost of product display and increases the variety of products choices⁸. It also provides “access” to the interior of complex devices, otherwise unavailable to see, creates virtual shop assistants or augments shelf displays⁹, and even creates virtual dressing rooms¹⁰.

This list may be extended with the recent interest in the area of VFX advertising¹¹. The idea is to extend an existing video content with product placement. The first step is to indicate the potential areas in the video that may hold media such as banners and posters. The second step involves using computer vision techniques, i.e., feature detection and tracking, region segmentation, optical flow, 3D and planar tracking to composite virtual advertisements into the footage in a photorealistic way.

Marketing and retail are constantly searching for new applications for augmented reality. One of the examples of interest in that matter, expressed by both academia and industry, is the 2016 AR hackathon organised by the University of California, Berkeley, with real-world use cases provided by Walmart Technology, and with support by Intel and Google (Project Tango). The hackathon’s participants were looking for possible answers to the question: “What can augmented reality achieve in the context of business and commerce?”¹².

While generally accepted as the desired novelty that attracts the customers, there is little research on measuring the impact of augmented reality on consumer decision making. Only recently, Hilken et al. [41] published a survey on the subject. They acknowledged AR as an emerging technology in retail, where, after the digital revolution, the shopping environment is changing to *multichannel* and to *omnichannel*. In multichannel, the offline and the online retail channels occur simultaneously, while in omnichannel, they are intertwined [10]. Therefore, “omnichannel” carries more potential for AR as it blends the boundary between the online and the offline. And vice versa, AR with its combination of real and virtual environments, makes a perfect tool for omnichannel shopping experiences.

Augmented reality is making its way into marketing and is present in more and more retail channels. People are getting used to it being present in their everyday shopping experiences. It may still not meet the high customer expectations, but progress is being made. AR, unlike VR, provides the customers with all the three attributes of an authentic and realistic experience [41]:

- *embedding* – all the information about the product can be retrieved at the same time when the decision is made,

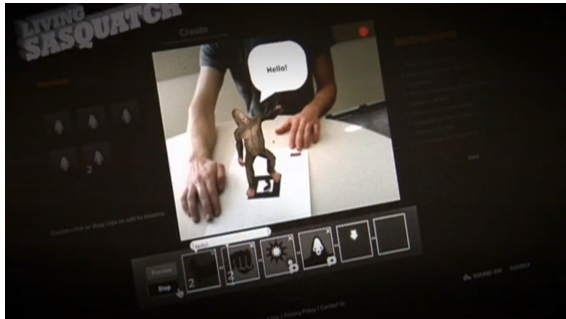
8 <https://blog.google/products/google-vr/bmw-i-and-tango-test-drive-new-app/>, last accessed 05/09/2019.

9 https://www.gmdc.org/sites/default/files/assets/pdf/white-paper_chapter-2.pdf, last accessed 17/03/2020

10 <https://venturebeat.com/2012/09/25/zugaras-virtual-dressing-rooms-take-the-x-factor-out-of-online-shopping/>, last accessed 17/03/2020

11 <http://www.mirriad.com>, last accessed 01/10/2019.

12 <http://tango.gplus.wtf/2016/12/28/what-can-augmented-reality-achieve-in-the-context-of-business-and-commerce/>, last accessed 06/10/2019.



(a) Customisable animated 3D objects



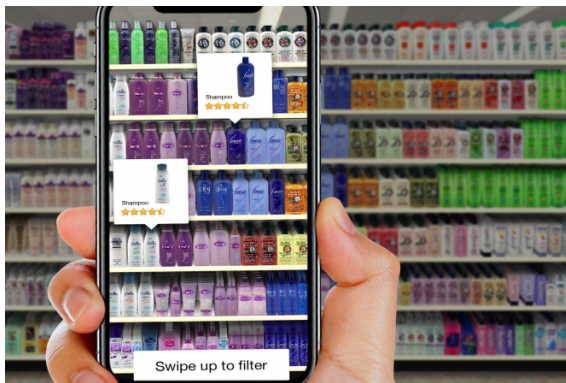
(b) Short 3D animations to build positive brand associations



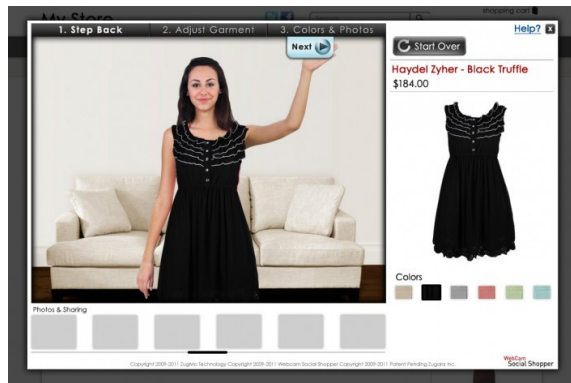
(c) AR car dealer showroom



(d) Augmented print



(e) Augmented supermarket shelves



(f) Virtual dressing room

Figure 2.9: Examples of applications of augmented reality to marketing and commerce.

- *embodiment* – customers can physically interact with a product,
- *extension* – they can communicate and share their experience with others.

While it is worthwhile to have these attributes in retail environments, it is even more desirable in market research environments, where study participants should behave as naturally as possible. That confirms that research on using augmented and mixed-reality solutions in market research, as presented in this dissertation, is worth pursuing.

Not all augmented reality platforms, though, are suitable for this project. As assets in market research are prepared in advance, based on carefully prepared scenarios, in my work I will focus on inserting virtual objects into already recorded videos. Objects should be inserted in real time and available for study participants online, which limits the available AR platforms mostly to computers and, less commonly, head-mounted displays. A game engine (in this case Unity) is a natural choice of environment for the project implementation because it can create content for both computers and HMDs (see Chapter 3, especially Section 3.1).

2.3 MIXED REALITY RENDERING

When considering realistic graphics insertions into real scenes, there are two main aspects that must be taken into account: correctly lighting the virtual objects to match the lighting conditions of the real scene, and the interaction between real and virtual objects, especially casting shadows, appearing in reflections and colour bleeding.

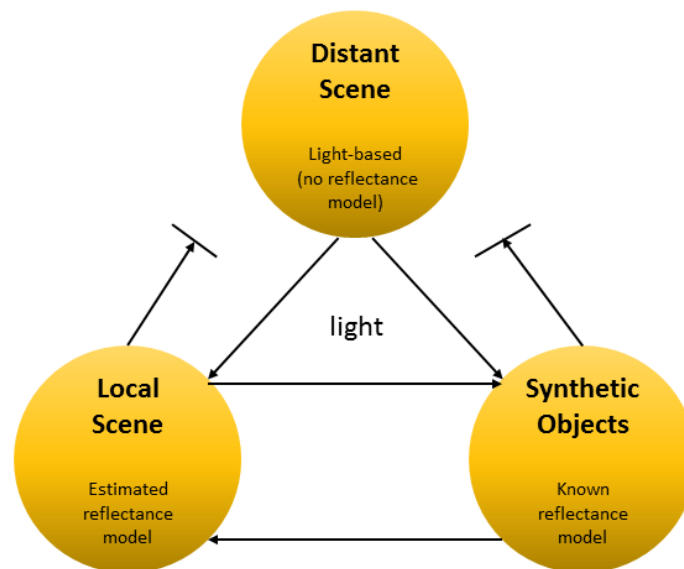


Figure 2.10: Debevec's light-based scene model [18].

2.3.0.1 *Light Transport in Mixed Reality Scenes*

To model interactions between real and virtual elements of a scene, Debevec introduced the light-based scene model [18, 19], which divides the scene into a local and a distant part (Figure 2.10). Only the local scene can interact with virtual objects, and light transport between them is done through differential rendering.

2.3.0.2 *Differential rendering*

Debevec’s differential rendering for light transport in mixed reality scenes allows the shadows cast on the simple virtual reconstruction of the local scene (e.g. a table top or ground plane) to be blended into the background image. In this technique, the background image, CG elements and the local scene reconstruction are rendered separately, and then the background image is updated with the difference between the empty local scene and the local scene with objects. The original description and illustration of this technique from Debevec’s paper [18] may seem confusing, as some elements appear only in the text, but are not explicitly illustrated. Kronander et al. [55] provide a more intuitive illustration (Figure 2.11) as well as a single formula for the final composite:

$$C = \alpha \cdot O + (1 - \alpha) \cdot (B + O - L), \quad (2.7)$$

where O is the image depicting inserted CG objects and the modeled local scene, L is the image with only local scene rendered, α is an alpha matte for virtual objects, B, C are the old background image, and the new background image updated with the difference between O and L .

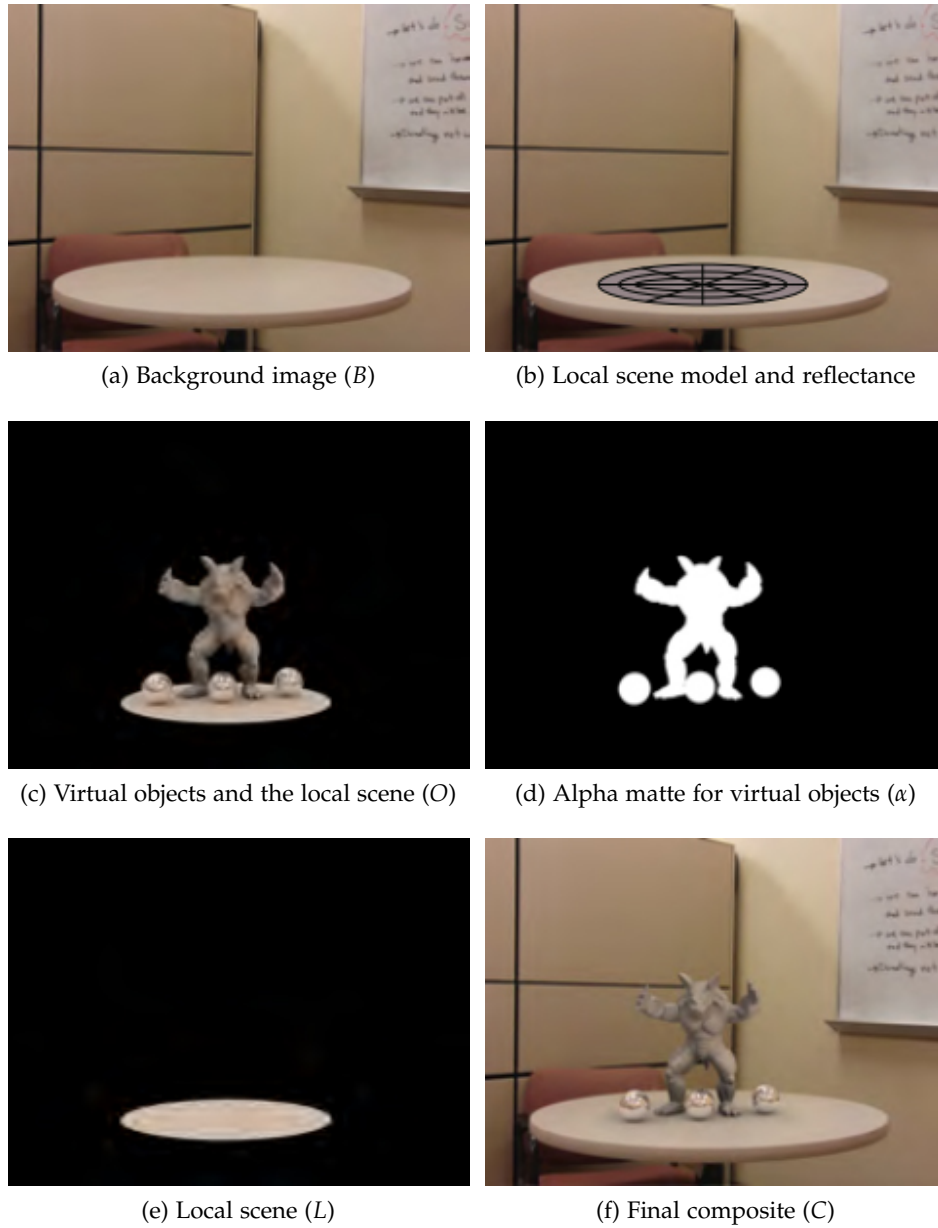


Figure 2.11: Image components of differential rendering: (a) the background image B , (b) the estimated model and reflectance of the local scene, (c) the virtual objects and the model of the local scene O , (d) the alpha matte for virtual objects α , (e) local scene rendered separately L , (f) the final composite C (Equation 2.7). Figure taken from Kronander et al. [55].

2.4 OMNIDIRECTIONAL CONTENT AND SPHERICAL CAMERA MODEL

Usually, when we think about a camera, we mean a perspective one, which we also call a *standard* camera. These are the cameras commonly used to take photos with, and this was the camera type I considered in Chapter 3. They are described by the pinhole camera model extended with the lens distortion model to compensate for image distortion caused by different types of lenses. Perspective cameras are designed to resemble human vision – without moving our heads, we can only see a section of the world in front of us.

Besides perspective cameras, there are other types of cameras, whose field of view is too wide to fit into the pinhole camera model even if extended with the lens distortion correction. They fall into the category of *omnidirectional* cameras:

“An omnidirectional camera (from omni, meaning all) is a camera with a 360-degree field of view in the horizontal plane or with a visual field that covers a hemisphere or (approximately) the entire sphere.” (Davide Scaramuzza [83])

Omnidirectional cameras found their use primarily in robotics as they outperform perspective cameras when used in navigation [84, 108]. Thanks to their wide field of view, they see the whole environment at once without the need to turn the camera around, and therefore, provide more landmarks for orientation in a single frame. That also makes navigation algorithms less likely to get lost in the case of a sudden turn.

The other applications of omnidirectional cameras include surveillance [116] and video-conferencing [16]. The reason to choose them over the standard cameras is the same: a wide field of view that shows the whole environment at once, whether it is a conference room or a surveilled building. In the past, these applications used mostly *catadioptric* cameras with a hemispherical field of view [72]. Catadioptric cameras consisted of a hemispherical mirror, which reflected the environment, and a standard camera recording this reflection. Recently spherical 360° cameras rapidly gained in popularity, making its way also into the media and entertainment industry.

Images taken with such wide-angled cameras have one characteristic that distinguishes them from perspective cameras. Instead of just providing a window into the world, they enclose the viewer, creating a new world around them. That attracted the attention of artists and documentary directors, who acknowledged 360° cameras as a powerful tool to create immersive experiences for their audiences.

2.4.1 Pre-digital Times

Immersion through the omnidirectional (or *panoramic*) images is a concept far older than digital displays or even cinematography. In fact, such images are older than photography itself. As far as in 18th century, there were painters who realised the benefits of immersing the viewer with their art, and creating an impression of being transferred somewhere

else. The artist who made the first panorama painting in 1792 was English painter Robert Barker. He also invented the term *panorama* itself, based on the Greek words *pan* (“all”) and *horama* (“view”). Barker’s cylindrical 360° paintings were displayed in a purpose-built rotunda and became a popular entertainment in Regency-time London and an inspiration to many of his followers.



Figure 2.12: London panorama of 1792, from the top of the Albion Mills, Robert Barker. *Image taken from Wikipedia.*

Another famous artist, Louis Daguerre, before he became the inventor of the first photographic process presented to the public, had been an equally successful creator of *dioramas* [113]. Dioramas require a special building for their display and consist of a huge painting, usually a landscape scene, that is partially translucent and could be backlit. The diorama show presented the painting changing its appearance. An example of such a change is turning a day scene into a night scene by changing the lighting. Daguerre was very creative in putting additional stimuli into his diorama shows, such as various types of light, sound, and moving mechanical parts, which we can compare to a “4D/5D/ n D” cinema today.

After the invention of photography, creators of panoramas naturally adopted the new medium in their work. The first devices that allowed a full 360° view to be captured and then displayed on a cylindrical screen were constructed by the Lumière brothers (1900), who called this new type of the image *photorama*. The photorama was very similar to modern photographic panoramas, which are combined from several photographs taken close to each other, but no stitching was necessary as the image was captured on one long piece of film (Figure 2.14). The projector used to display the photorama (Figure 2.15) bears a striking resemblance to modern 360° cameras rigs such as Google Jump [3] (Figure 2.17) or the rig designed by Disney Research [86], and the camera that was able to capture the panoramic image contained a rotating lens (Figure 2.16), a technique that recently re-appeared in state-of-the-art 3D panoramic video streaming [54].



Figure 2.13: The last existing diorama painted by Daguerre in 1842, located in the church in Bry-sur-Marne, France. It creates a cathedral-like spatial illusion on the wall of a small church, and brings together two realities, physical and imaginary, in a similar way to mixed reality systems nowadays which achieve this through digital devices. *Image taken from the town of Bry-sur-Marne website*¹³.



Figure 2.14: Lumière's photorama view. The cylindrical panoramic photo was captured on one long piece of film by a special camera with a rotating lens. The only existing example can be viewed in the Institut Lumière in Lyon, France. *Image taken from the Institut Lumière website*¹⁴.

¹³ www.bry94.fr/Eglise-Saint-Gervais-Saint-Protas-et-le-Diorama-de-Daguerre.html (French)

¹⁴ <http://www.institut-lumiere.org/musee/les-freres-lumiere-et-leurs-inventions/photoramas.html> (French)



Figure 2.15: Panoramic projector for photoramas. *Image taken from the Institut Lumière website.*



Figure 2.16: Camera used to capture photoramas. The handle on the top rotates the lens around the cylinder, which allows the camera to capture the full 360° panorama. *Image taken from the Institut Lumière website.*



Figure 2.17: Google Jump VR camera rig. *Image taken from Anderson et al. [3].*

2.4.2 Spherical Camera Model

As defined at the beginning of this chapter, *omnidirectional* is a term used to describe a range of cameras. Due to their high image distortion caused by the extremely wide field of view, they cannot be described by a pinhole camera model. Nowadays, however, *omnidirectional* has become a synonym for 360° spherical cameras. This is due to their rapidly increasing popularity, particularly in the entertainment sector. There is a variety of cheap consumer 360° cameras on the market, so the price ceased to be a barrier. Therefore, one should be aware that, even though the most recognisable of the type, spherical cameras are only a subclass of omnidirectional cameras.

A spherical camera is able to capture images with 360° horizontal and 180° vertical (full spherical 4π steradian) field of view. To be able to do that, it requires a special geometric model, which is illustrated in Figure 2.18. The spherical camera was defined by Torii et al. [105] as a pair: a unit sphere and its center C , equivalent to the camera center. It also has its orientation in space, defined by the forward vector \mathbf{f} , which lies on a great circle of the sphere and is perpendicular to the camera up axis \mathbf{u} . The up axis is, in turn, defined by the line from the nadir N (the bottom-most point on the sphere) to the zenith Z (the top-most point on the sphere), passing through the camera centre.

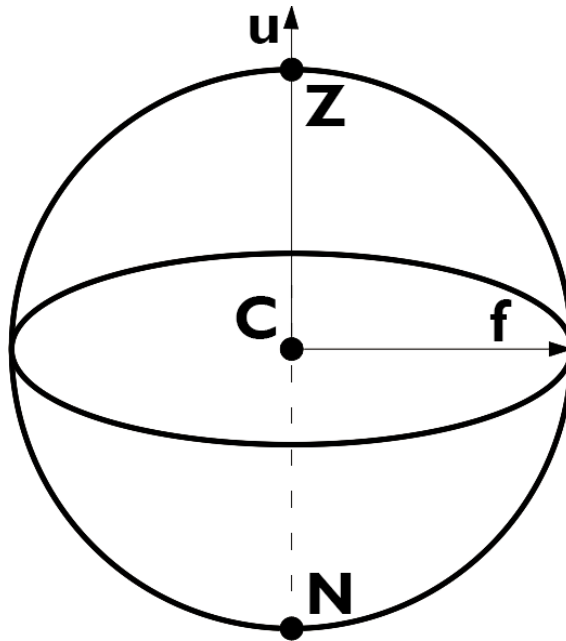


Figure 2.18: The spherical camera model consists of a unit sphere and its centre C . Its orientation in space is set by the forward vector \mathbf{f} and the up vector \mathbf{u} . The up vector lies on the line connecting two extreme points on the sphere, the zenith Z on the top and the nadir N on the bottom.

The spherical camera is a purely virtual concept as it does not exist physically. It is because there is no single device with optics able to capture a full 360° field of view. Every physical lens, with no matter how wide a field of view, has a blind spot directly behind it, where the image is captured. Therefore, practical implementations of the spherical camera use a set of at least two calibrated fisheye or more perspective cameras with overlapping fields of view. The images from the cameras are then stitched to emulate the 360° field of view.

2.4.3 360° Image Representations

360° images capture the world around the camera and thus they form a 3D sphere encapsulating the viewer. However, they are created from a series of standard 2D images warped on a sphere, and therefore they can be unwarped back to 2D pixel representation for storage and image processing.

The most popular 360° image representations are directly inherited from types of mapping for environment maps used in image-based lighting [18], which in turn are derived from cartography, as they share the same geometry. These are the latitude-longitude (lat-long, or in case of 360° imagery more commonly known as equirectangular), the angular (or spherical), and the cubemap mapping. They were recently complemented with a few new mapping modes associated with new ways of acquiring 360° images (Figure 2.19d) or their display, especially for streaming 360° video (Figures 2.20 and 2.21). The last two mappings are new propositions by Facebook and Google, which aim to reduce processing time when displaying the image inside the head mounted display and to map solid angles more uniformly to pixels.

Below I compiled a list of names and corresponding examples of these mappings in an attempt to clarify their naming conventions, as some of them are often confused:

- Angular/spherical (often confused with spherical panorama (Figure 2.19a)).
- Cubemap (Figure 2.19b).
- Equirectangular/latitude-longitude(lat-long, LL)/panoramic/spherical panorama (Figure 2.19c).
- Dual fisheye (two half-spheres), associated with a particular spherical camera geometry with two lenses and a mirror, such used in the Ricoh Theta series (Figure 2.19d). I included it in this list as there were some attempts to use this representation in finding correspondences between the consecutive video frames [64], but most of the two-lens 360° cameras offer already stitched images in equirectangular format, without access to dual fisheye intermediate image.

- The Pyramid format introduced by Facebook [29] aimed at video streaming to reduce bandwidth. The invisible part of the image behind the viewer is streamed in lower resolution than the visible part in front of the viewer (Figure 2.20).
- Equi-Angular Cubemap introduced by Google [35] for video streaming. In this format, pixels cover equal areas of the image to reduce distortion, which works better for video compression. Also, this mapping distributes pixels more evenly than traditional equirectangular projections, where the poles are represented by more pixels than the equatorial band, even though it contains the most important content (Figure 2.21).



Figure 2.19: The most popular omnidirectional mapping representations.

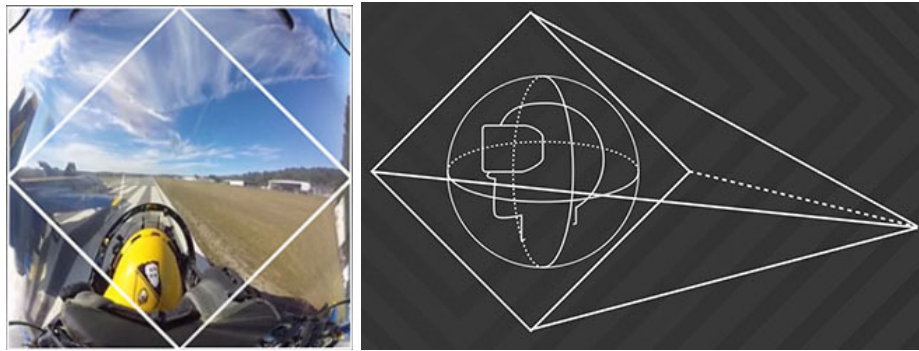


Figure 2.20: The Pyramid format for 360° video streaming developed by Facebook. *Image taken from Facebook [29].*

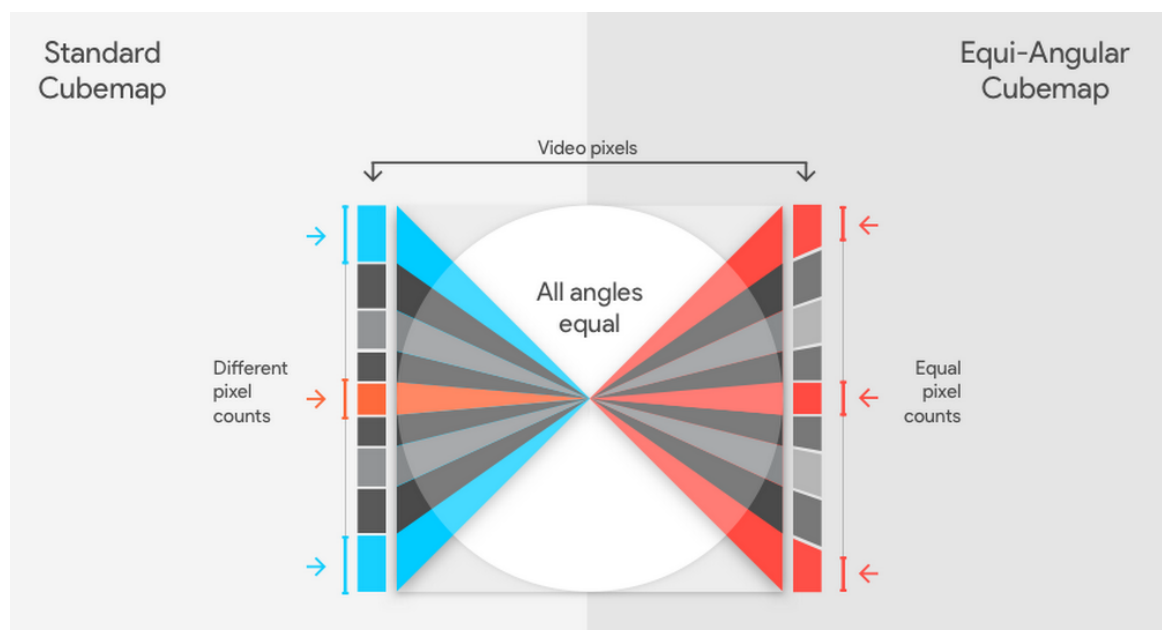


Figure 2.21: The Equi-Angular Cubemap format for 360° video developed by Google compared with a standard cubemap on the left. *Image taken from Google [35].*

Unless inside a spherical dome display, native 360° images cannot be viewed on a standard screen in their spherical form. Even VR headsets display the content on two 2D screens, one for each eye. Therefore, viewing 360° images requires mapping them to the perspective view as that of a standard camera. In other words, they should be rendered by a supplementary virtual perspective camera, placed in the centre of the image sphere. Figure 2.22 shows an example of the equirectangular image mapping and a section of it rendered as a perspective view by a virtual camera.



Figure 2.22: **Left:** The equirectangular representation of a 360° spherical image. **Right:** A perspective view of the part of the image.

2.4.3.1 Mapping Between Image Representations

Every omnidirectional image mapping mentioned above originates from a spherical image, thus for every such representation exists a formula that maps the pixel values from the 2D image domain to a 3D unit sphere. And vice versa, there is an inverse mapping formula from a sphere to a 2D image representation. All these formulas use conversion to spherical coordinates (φ, θ) as an intermediary step and take as an input normalised image coordinates (u, v) . Figure 2.23 illustrates the relation between normalised image coordinates and spherical coordinates for equirectangular mapping.



Figure 2.23: Normalised image coordinates (u, v) and spherical coordinates (φ, θ) in equirectangular mapping.

Below are the three most common mapping formulas for 360° spherical images. The vector $[D_x, D_y, D_z]$ represents a point on a unit sphere and the vector $[u, v]$ represents a point in normalised image coordinates. Note that all these formulas depend on the chosen coordinate system.

FORWARD AND INVERSE LATITUDE-LONGITUDE MAPPING

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \begin{bmatrix} \sin \theta \sin \phi \\ \cos \theta \\ -\cos \theta \sin \theta \end{bmatrix}, \quad \begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} 2\pi u \\ \pi(v-1) \end{bmatrix} \quad (2.8)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 + \frac{1}{\pi} \arctan(D_x, -D_z) \\ \frac{1}{\pi} \arccos D_y \end{bmatrix} \quad (2.9)$$

Latitude-longitude mapping, or more commonly known as *equirectangular* when referring to spherical images, is the base mapping used in the context of my project. It is the most popular output from consumer 360° spherical cameras, which produce already stitched 360° images and videos. I use the equirectangular mapping to convert tracked feature points from image domain to 3D coordinate system (Section 5.1.1), where they can be further used in omnidirectional epipolar geometry to retrieve the camera pose (Section 5.1.2), and in triangulation to reconstruct 3D points (Section 5.1.2.1). Mapping from image coordinates to 3D sphere and back is also a part of image stabilisation by the de-rotation procedure I describe in Section 6.1.1.

FORWARD AND INVERSE ANGULAR MAPPING

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \begin{bmatrix} \cos \theta \sin \phi \\ \sin \theta \sin \phi \\ -\cos \phi \end{bmatrix}, \quad \begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} \arctan(1-2v, 2u-1) \\ \pi \sqrt{(2u-1)^2 + (2v-1)^2} \end{bmatrix} \quad (2.10)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{2} + \arccos\left(\frac{-D_z}{2\pi \sqrt{D_x^2 + D_y^2}}\right) \begin{bmatrix} D_x \\ D_y \end{bmatrix} \quad (2.11)$$

This mapping is applied in old methods of retrieving environment maps for image-based lighting by photographing a mirror ball. It is implemented in software I used to process a series of photographs to produce an HDR environment map [21]. Therefore, the angular mapping method is implicitly used in lighting objects inserted into perspective videos, in the first part of my project (see Section 3.1.6).

FORWARD AND INVERSE CUBE MAPPING

In the cube mapping, each of six sides of a cube is mapped separately. The mapping

formula for one (middle) side (see Figure 2.24) is as follows, and the formulas for other sides are analogous:

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \frac{1}{\sqrt{1 + (2u - 1)^2 + (2v - 1)^2}} \begin{bmatrix} 2u - 1 \\ 2v - 1 \\ 1 \end{bmatrix}, \text{ if } u \in \left[\frac{1}{3}, \frac{2}{3}\right] \text{ and } v \in \left[\frac{1}{2}, \frac{3}{4}\right] \quad (2.12)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{3}{2} \begin{bmatrix} \frac{-D_x}{2D_z} \\ \frac{D_y}{2D_z} \end{bmatrix}, \text{ if } (D_z < 0) \wedge (D_z \leq -|D_x|) \wedge (D_z \leq -|D_y|) \quad (2.13)$$

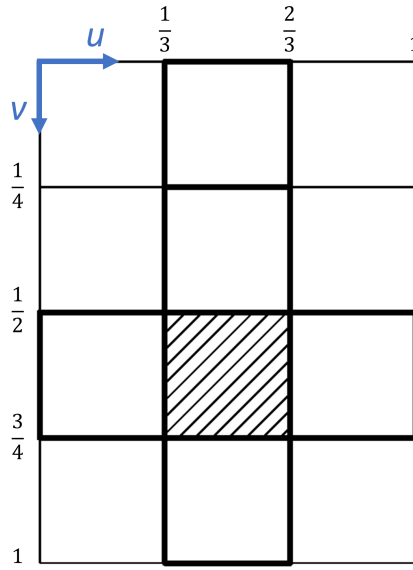


Figure 2.24: Normalised image coordinates (u, v) in cube mapping. The hatching marks the middle side for the mapping formula above.

Cube maps are used by many graphics software programs as internal representations to store and process environment maps for image-based lighting. This is also true for the Unity game engine, which provides the rendering environment for my project. Depending on the input format (angular mapping in case of traditional mirror ball photographs and equirectangular mapping in case of 360° images), imported environment maps are converted to cube maps for further use.

Every 2D omnidirectional image representation maps to a sphere, and also exists an inverse mapping from a sphere to an image representation, therefore it is possible to map from one image representation to another, using mapping to a sphere as an intermediate step. Sometimes it is necessary, as different applications require different image representations.

2.5 OMNIDIRECTIONAL STRUCTURE FROM MOTION

Structure from motion (SfM), sometimes called *multi-view reconstruction*, originated from photogrammetry (literally meaning “measuring from photos” in Greek), which is a technique used in geodesy and cartography to retrieve 3D terrain models from the aerial photographs [115]. The goal of SfM is to reconstruct the environment and to calculate the camera motion given a set of image correspondences, first on a frame to frame basis, and then using the whole set of images in a global optimisation called bundle adjustment. SfM is based on epipolar geometry, which describes the relationship between the two different camera views based on their image correspondences [40].

2.5.1 Theoretical Foundations for Omnidirectional Structure from Motion

One of the earliest works on 3D reconstruction from 360° images was by Kang & Szeliski [50]. They proposed using cylindrical panoramas to extract 3D depth from the scene to avoid merging errors associated with depth maps retrieved from a set of perspective images. Chang & Hebert [13] were the first to formulate the epipolar geometry problem for 360° cameras. However, in their paper, they discuss only the catadioptric camera model and confine themselves to estimating the camera pose without reconstructing the scene. Torii et al. [105] explained the foundations of this problem in more details and defined a general spherical camera model as consisting of a camera centre—the point in space where all viewing rays intersect—and the surface of a unit sphere surrounding the centre. They formulated two- and three-view geometry for spherical cameras by analogy to their pinhole equivalents. Fujiki et al. [31] generalised the problem to a pair of cameras placed arbitrarily in space. However, it is convenient to consider, without loss of generality, the local coordinate system of one of the cameras to be aligned with the world coordinate system.

2.5.2 Image Correspondences

Every SfM or SLAM algorithm begins by finding image correspondences. For basic proofs of concept, it might be sufficient to use synthetic [13, 31] or manual correspondences [13, 64]. For practical scenarios, a variety of feature descriptors have been proposed, including BRISK [58], Accelerated KAZE (A-KAZE) [2] and Affine SIFT (ASIFT) [76]. While these descriptors were originally designed for standard projective cameras, they have also proven to perform well for 360° images [6, 77]. They also outperform the popular SIFT descriptor when applied to equirectangular images [6, 61]. Spherical SIFT (SSIFT) [15] and Spherical ORB (SPHORB) [120] introduce spherical versions of popular planar feature descriptors.

For longer videos, computing and matching feature descriptors across all frames quickly becomes expensive. In addition, if a scene contains many repetitive elements, the number

of mismatched features increases dramatically. I track features over multiple frames to speed up execution and reduce mismatched repetitive features.

2.5.3 *Feature Tracking*

Im et al. [45] used the KLT tracker [63, 87, 104] separately on the two fisheye images of a commodity 360° camera. Their implementation of bundle adjustment reflects this input data model, with a cost function divided into two components associated with front and back views. However, this approach does not work for longer videos or videos with substantial camera rotation, as features are not tracked when they move from the front to the back view and vice versa, which results in shorter feature trajectories. My implementation treats the stitched image from the camera as a sphere and therefore does not fail to track arbitrary camera rotation.

However, the majority of approaches apply standard tracking techniques to the input perspective views of 360° images before stitching, or to 360° images projected onto a cube map, where each face of the cube represents the image of a virtual perspective camera. Michiels et al. [69] performed tracking on undistorted images from separate cameras on a 360° multi-camera rig, and thus protected the solution from stitching errors, but their method requires feature matching between the cameras in every frame which loses the main benefits of the tracking approach. Huang et al. [43] used projection on slightly overlapping cube map sides to reliably track features near side edges, but only for a small camera movement between the frames. The most popular commercial tracking tools, such as Spherical Tracking Toolset for PFTrack [103], Cara VR plugin for Foundry Nuke [30] and Mettler SkyBox Studio V2 plugin for Adobe After Effects [68] also adopt the perspective approach, building on their existing 3D tracking pipelines for standard perspective cameras.

2.6 REAL-TIME PHOTOREALISTIC GRAPHIC INSERTIONS INTO 360° VIDEOS

One goal of object insertion is for augmented reality or mixed reality: the fusion of physical and virtual realities when seen through an intermediary device such as a monitor, mobile device, or head-mounted display [5]. Section 2.2 defines both AR and MR, and gives the reader some insight into their history and applications. Here, I focus only on the topics related to AR/MR and 360° videos.

2.6.1 *Illumination Estimation*

Recovering the illumination of the environment allows virtual objects to be rendered under matching illumination [55]. Light field transfer provides the highest quality results [14], but is complex to capture. In practice, image-based lighting based on environment maps

(or *light probes*), described in Section 2.3, has much lower complexity, and has been shown to provide high-quality results under infinitely-far-away scene assumptions.

Many different approaches have been proposed for estimating the illumination in a scene—a problem known as *inverse lighting*. If there are few distinct light sources, we can estimate their locations and colours directly [26, 62]. However, real-world illumination is generally more complex and estimating it requires some assumptions about the scene. If the geometry of the scene is known, for example if it was modelled manually [51] or reconstructed from an RGB-D camera [38, 81, 118], we can optimize for the environment map which best explains the input images or videos. Other priors may be useful, for example knowing that the input contains a face [11, 53], an indoor [34] or outdoor scene [42, 57], or multiple instances of the same object [112]. However, the most detailed illumination is the one captured directly by 360° cameras [46, 69, 80, 119], and this is the approach I chose. It does not require any additional effort when recording the video as it utilises the video itself.

Most cameras—360° or not—do not capture imagery with high dynamic range (HDR). However, HDR image-based lighting improves visual results by overcoming muted low-contrast reflections [18, 19] (see Figure 6.3). Iorns & Rhee [46] introduce basic inverse tone mapping, which they apply to low-dynamic-range 360° video and obtain satisfactory results, in some cases comparable to lighting the scene with HDR images. Their approach works in real time and produces results that can be viewed in a VR headset. Rhee et al. [80] further extend this work using image-based shadowing, which adds a drop shadow based on the brightest detected light source, i.e., usually the sun in outdoor environments. They demonstrate high-fidelity results from low-dynamic-range 360° videos in real time. Recent progress in deep learning has resulted in multiple concurrent techniques for inverse tone mapping [25, 27, 60, 67, 82], which reconstruct HDR imagery from standard (low-dynamic-range) imagery. However, most of them use masks of oversaturated pixels in their pipelines [25, 60, 82]. Therefore, they perform inpainting of oversaturated areas rather than inverse tone mapping and fail to recover HDR versions of well-exposed LDR images. This rules these methods out when considering the conversion of 360° video to HDR for improved image-based lighting results.

2.6.2 Real-time 360° Virtual Object Insertion

The most similar virtual object insertion approach to mine is that of Michiels et al., who used 360° imagery for object insertion with user interaction [69]. Their approach renders reflections according to the position of the viewer and based on material properties of the inserted object. They use precomputed radiance transfer to achieve real-time physically-based global illumination. However, they appear to use 360° images directly as environment maps without first linearising the colour space or recovering HDR illumina-

tion information. Their pipeline also lacks shadowing. These two factors limit the visual fidelity of their results.

Rhee et al. [80] presented a complete pipeline for inserting virtual objects into 360° videos with real-time user interaction and image-based shadowing but only for static cameras.

DYNAMIC MIXED-REALITY COMPOSITING

Fully computer-generated environments used to create assets for market research surveys can produce multiple study scenarios in real time if pre-programmed accordingly. The aim of this project is not only to provide a more realistic replacement for these environments but also to match their flexibility in this regard. This can be achieved by adding changeable CG elements to the real background with digital compositing, which allows the two images to be blended on a pixel-by-pixel basis.

What I acknowledge as the *standard way of compositing*, is combining different types of 2D graphical assets off-line in postproduction [78]. These are still images (such as photos or renderings) or videos and rendered animations (see Chapter 1, and especially Section 1.1). In this form, compositing is not easily adaptable to even minor changes in source CG elements, as they need to be re-rendered. Thus, standard compositing fails in applications that demand a more flexible approach, such as creating multiple assets for market research surveys.

In particular, as it is in case of conjoint analysis, when many versions of the same scene are required, with the same background plate, but different 3D objects, or even with the same 3D objects but with different textures. It seems to be an unnecessary waste of resources to render each version of the whole scene separately and composite them onto the background, while most of the visual data is redundant.

This chapter presents a dynamic mixed-reality compositing solution to this problem as opposed to the standard compositing. It allows the live action footage to be displayed only once as a background and to render CG elements on top of it when needed. In this dynamic approach, only the camera and sparse scene reconstruction (the output from matchmoving) is required together with 3D models of the objects intended to insert into the scene.

As mentioned earlier, mixed-reality compositing techniques found their uses in virtual cinematography and on-set previsualisation, reducing the cost of changes made further in the postproduction pipeline. The use of a game engine as an environment for on-set previsualisation was first proposed by Northam et al. [75], who combined it with a motion capture system for visualising multiple characters in one scene. Since then, game engines have established themselves as a standard tool in modern film production. The most

popular commercial on-set camera tracking and previsualisation systems [59, 73, 93] all offer integration with game engines, usually both Unity [109] and Unreal Engine 4 [28].

One of the most spectacular recent examples of on-set previsualisation is The Mill's Blackbird vehicle rig [107], which allows any type of car to be visualised in place of this custom-built rig. It combines real-time motion tracking with rendering in a game engine, and uses Google Tango augmented reality technology to customise the appearance of the rendered car. Multiple light probes attached to the rig provide environment maps for image-based lighting¹.

However, to the best of my knowledge, there is no example in the literature of using a game engine with a pre-tracked camera for compositing purposes, as I will present next.

3.1 DYNAMIC MIXED REALITY COMPOSITING PIPELINE

I chose the Unity game engine as an environment for the implementation of my dynamic compositing pipeline, and the reason was twofold. Firstly, a need for real-time rendering tools and interaction with the inserted objects implied a game engine as the only environment that meets these requirements. Secondly, I chose this particular game engine because it was already a part of the asset creation pipeline of my host companies.

Since Unity has been designed as a game engine, and not as a compositing tool, many of its features are optimised for real-time game performance, which differs in its principles from the requirements of a postproduction pipeline. Therefore, the biggest challenge that my dynamic mixed-reality compositing faces is to provide the implementation for an interface between the input data and the way they are handled and displayed inside the game engine.

My main contribution is a practical solution for synchronising clock-driven and event-driven elements in the Unity game engine environment, to make correct graphics element insertions into live-action footage possible. I present a complete pipeline for dynamic mixed-reality compositing that provides users with tools for interacting with inserted elements, manipulating them in real time, and creating any number of different versions of the same scene in a dynamic way. My pipeline is completed by differential rendering for image-based shadowing of inserted objects.

Figure 3.1 illustrates the proposed dynamic compositing pipeline with Foundry Nuke for camera tracking and sparse scene reconstruction, 3ds Max for exporting camera animation and creating 3D objects and Unity for combining all elements. The Unity engine was chosen as an environment for interaction with the elements of a scene, because within it, the CG elements can be adjusted or moved to another position, different textures and materials, lighting types, positions and colors can be tried during playback, without

¹ <https://www.theverge.com/2017/3/1/14778662/epic-games-chevrolet-the-mill-cyclops-car-motion-capture>

the necessity to render them again after each change, and to re-import the footage and adjusting the whole scene in the compositing software.

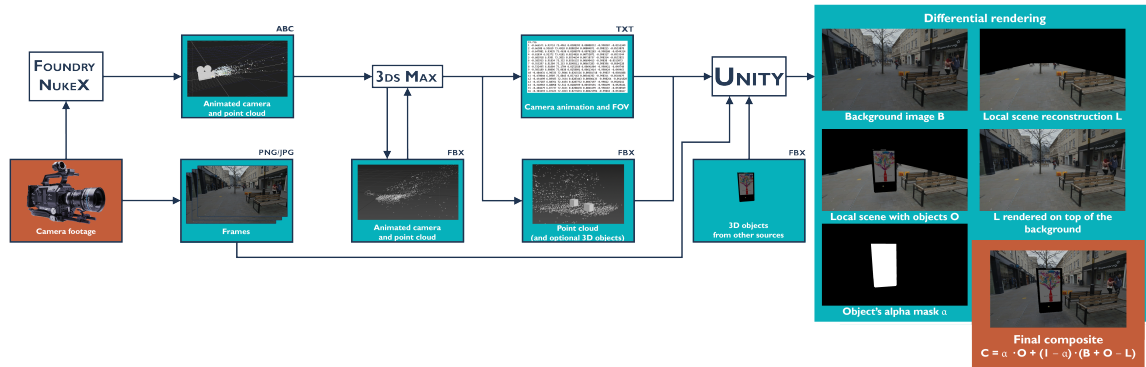


Figure 3.1: Dynamic compositing pipeline with Foundry Nuke for camera tracking and sparse scene reconstruction, 3ds Max for exporting camera animation and creating 3D objects, and Unity for combining all elements to produce the final composite.

3.1.1 Input Data Formats

The first step in the pipeline is 3D camera tracking and sparse scene reconstruction performed by Foundry Nuke. Then, I export the animated camera pose and reconstructed point cloud to the Alembic (ABC) file format that is commonly used in visual effects applications. I chose it over the more popular FBX format used in 3D modelling software due to its insensitivity to resampling animation curves when converting between different coordinate systems². Such an operation causes inaccuracies in camera rotation. This is not the case with ABC files, because they store “baked” animations, which are independent of axis directions and rotation. However, ABC files are currently not supported by Unity and create their own camera object when imported into 3ds Max, which is different from native 3ds Max cameras as it is non-editable. This is because the baked nature of the animation inside the Alembic file does not allow any changes to be made to the original values.

I overcome these shortcomings by using 3ds Max as a conversion tool between the ABC and FBX formats. When exported to an FBX file and imported back, a camera appears as a standard 3ds Max dummy object, and a new camera can be created, aligned and linked with it. Unfortunately, the focal length and film back size still need to be set manually.

² http://download.autodesk.com/us/fbx/20112/3dsmax/_index.html

3.1.2 Coordinate Systems

Different software uses different coordinate systems, and it is important to perform correct conversions between them. Rotation, in particular, requires special attention because, being expressed as a matrix multiplication, and therefore non-commutative, depends on the axes order and coordinate system handedness. In our case, three coordinate systems associated with the software used in the compositing pipeline should be taken into account (Figure 3.2).

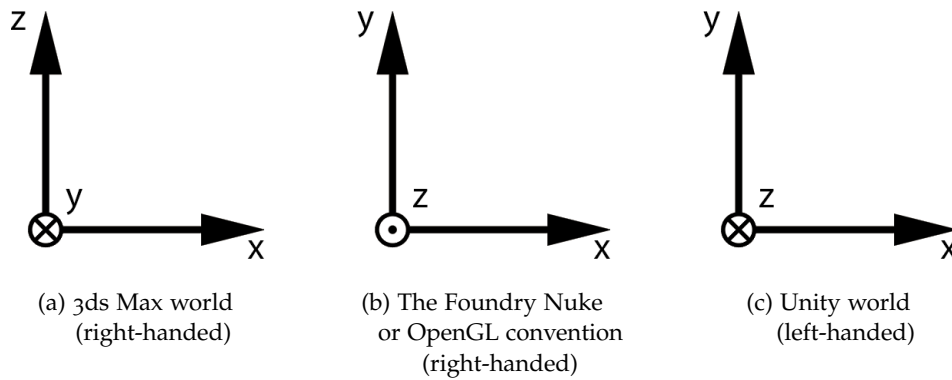


Figure 3.2: Different coordinate systems used by the software in the compositing pipeline. \odot represents an axis pointing towards the viewer, and \otimes represents an axis pointing away from the viewer, both perpendicular to the image plane.

When a virtual camera is set up to correspond to a real camera, its position and rotation in every animation frame is exported to a text file by my custom script. It performs conversion between 3ds Max and Unity coordinates systems according to the following steps:

1. Conjugate the rotation quaternion to change the direction of rotation (for the opposite handedness of the target coordinate system).
2. Convert the quaternion $\mathbf{q}_{\text{Max}} = (q_x, q_y, q_z, q_w)$ to a rotation matrix \mathbf{R}_{Max} .

3. Convert the 3ds Max coordinate system to the OpenGL coordinate system by rotating it by 90° around the x-axis, which is equivalent to multiplying the rotation matrix and translation vector by the following matrix:

$$\mathbf{T}_{\text{Max} \rightarrow \text{OGL}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

$$\mathbf{R}_{\text{OGL}} = \mathbf{T}_{\text{Max} \rightarrow \text{OGL}} \cdot \mathbf{R}_{\text{Max}} \quad (3.2)$$

$$\mathbf{t}_{\text{OGL}} = \mathbf{t}_{\text{Max}} \cdot \mathbf{T}_{\text{Max} \rightarrow \text{OGL}} \quad (3.3)$$

Where \mathbf{R}_{OGL} , \mathbf{t}_{OGL} and \mathbf{R}_{Max} , \mathbf{t}_{Max} are 4×4 rotation matrices and 1×4 translation vectors in OpenGL and 3ds Max coordinate systems respectively.

4. Change coordinate system handedness from left-handed to right-handed by changing the direction of the z-axis:

$$\mathbf{R}_{\text{Unity}} = \mathbf{S}_z \cdot \mathbf{R}_{\text{OGL}} \cdot \mathbf{S}_z \quad (3.4)$$

$$\mathbf{t}_{\text{Unity}} = \mathbf{t}_{\text{OGL}} \cdot \mathbf{S}_z \quad (3.5)$$

where \mathbf{S}_z is a transition matrix [24]:

$$\mathbf{S}_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

5. Convert rotation back to quaternion.
6. Swap the q_y and q_z components of the quaternion $\mathbf{q} = (q_x, q_y, q_z, q_w)$ and negate q_x and q_y to match the rotation order in Unity:

$$\mathbf{q}_{\text{Unity}} = (-q_x, q_z, -q_y, q_w). \quad (3.7)$$

3.1.3 Camera Model

Unity uses the simplest camera model, which is a pinhole camera with only one intrinsic parameter, vertical field of view (FOV). This model ignores film back size and focal length, which are implicit in FOV. If their values are known, horizontal and vertical FOVs can be calculated using the following formulas:

$$\text{horizontal FOV} = 2 \arctan \frac{w}{2f_x} \quad (3.8)$$

$$\text{vertical FOV} = 2 \arctan \frac{h}{2f_y} \quad (3.9)$$

where w and h indicate the size of the film back (width and height, respectively) and f_x, f_y are focal lengths (usually $f_x = f_y$ for square pixels). The camera aspect ratio is by default the aspect ratio of the screen if not changed in script.

In my pipeline, vertical FOV is calculated in 3ds Max, which allows the real camera parameters mentioned above to be used explicitly. Then the vertical FOV value (as required by Unity) is saved in the exported text file together with the camera animation.

I assume that the lens distortion has been corrected beforehand and, therefore, does not need to be modelled in Unity. Otherwise, any desired lens distortion could be applied using a custom image post-processing effect.

3.1.4 Synchronisation

One of the most notable differences between games and film production pipelines is the type of frame rate. In the case of games, it is variable and depends on the current processor and graphics card load as well as general device performance. In addition, games require much higher display frame rate than movies to run smoothly, with more than 60 fps (frames per second) or higher. On the other hand, video clips and animations are recorded with a constant frame rate, usually 24, 25 or 30 (29.97) fps. Real-time game engines such as Unity attempt to preserve this frame rate by using an internal clock to calculate the frame number of an animation or a video to be displayed whenever the game update function is called. However, this does not guarantee an exact frame rate to be matched, only approximated. For the human eye, the change of speed is hardly noticeable when playing. The frame rate of standard animations or videos is also significantly lower than usual game requirements. To compensate for this mismatch, Unity by default interpolates incoming animation curves between the original frames to make them smoother at higher frame rates.

This interpolation makes it difficult to exactly synchronise camera animation and video footage. In this section, I describe how I approach this challenging and non-trivial task.

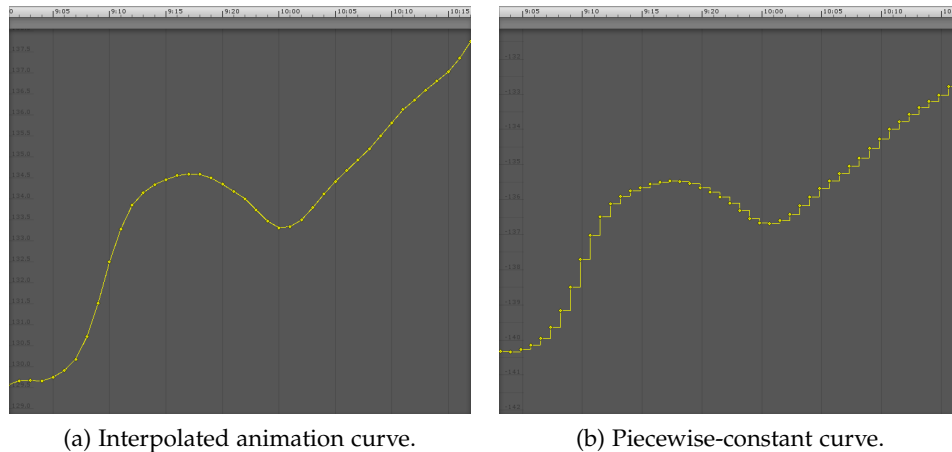


Figure 3.3: (a) Unity automatically interpolates animations to smoothly render them at higher rendering frame rates. (b) The motion of the virtual camera should not be interpolated, but piecewise constant, to synchronise the camera pose to each displayed video frame throughout its duration.

3.1.4.1 Camera Animation in Unity

I start by tracking the camera pose in my input footage using the 3D camera tracker built into Foundry Nuke. The per-frame camera pose can be imported into Unity as a part of the FBX file and is interpreted as an animation that can be applied to any game object, in particular to a virtual camera as camera objects do not transfer directly between FBX files and Unity. Animations can be viewed in the animation editor as a set of animation curves, one for every position and rotation dimension, determined by the key frames. As mentioned earlier, these curves are resampled between key frames to make transitions smoother at higher frame rates. However, this creates interpolated camera poses with no corresponding video frame, which inevitably leads to synchronisation problems.

To prevent the camera animation curve from being resampled, one should disable curve resampling as well as compression during import of the curve file. To additionally ensure that there are no in-between values, even when resampled, animation curves should be converted into a piecewise constant curve (see Figure 3.3) by fixing its value between the original keyframes. Such a transformation is possible in Unity on a copy of the animation that is detached from the rest of the original FBX file. However, that causes occasional problems with setting correct rotation angles on either side of a keyframe, which brakes the continuity of rotation in that point and results in the camera facing unexpected directions. Therefore, it is advisable to create the desired curve shape in the 3D modelling software of choice before exporting it to the FBX format.

3.1.4.2 Video Clips in Unity

The release of Unity 5.6 (March 2017) introduced a new feature for playing back video clips, called *Video Player*. It replaces the obsolete *Movie Texture*, which was not supported on every platform. Video Player allows the video to be rendered directly as the camera's background, without having to texture map the video onto an existing object. Video Player also offers automatic video playback, so no additional scripts are required to control the video, as was in the case before, with *Movie Texture*.

3.1.4.3 Initial Synchronisation Attempt

As mentioned earlier, the playback of animations and media is driven by the internal clock of the game engine. That means that any duration in frames is automatically interpreted as a duration in seconds based on the frame rate associated with the media asset. To select a specific video frame, the frame index thus has to first be converted to the corresponding time stamp. The synchronisation is therefore performed in the time domain.

I wrote a script that synchronises camera pose animation and video as follows: whenever the game window update function is called, it sets the playback time of the animation to match that of the video. This way, it allows them to be adjusted between consecutive update calls in real time, as changing camera pose is computationally inexpensive. The reverse way of synchronisation, where video playback time is matched with the animation time, causes delays in displaying the video content as seeking for a frame in a video can be expensive.

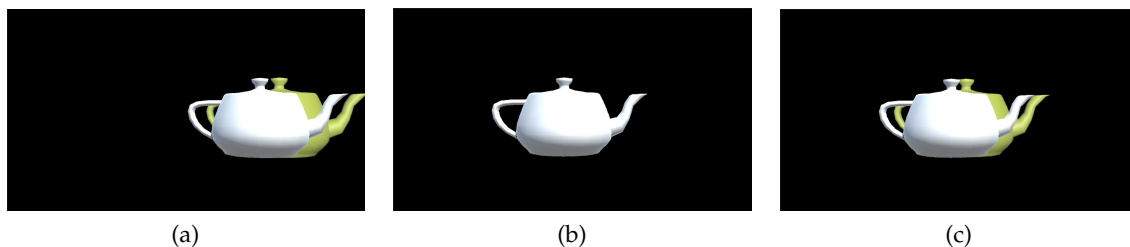


Figure 3.4: Frames rendered by the Unity virtual camera (a white teapot object placed in the scene) with the ground truth animation in the background (a yellow teapot image). While in some frames both views match each other (b), there are frames where rendered video and Unity camera motion are not synchronised (a,c).

I tested my synchronisation script on the toy example in Figure 3.4. I set up a moving virtual camera which rendered a scene with a yellow teapot as a video, imported it into Unity, where it is overlaid by a white teapot with the same size and camera motion. If the synchronisation is successful, the rendered image of Unity's white teapot would be perfectly aligned with its yellow image in the video.

Unfortunately, this test demonstrated that the animation curve appears to still be resampled when playing, despite this being disabled during import (see above). This is a well-known, and at the time of writing this dissertation still officially unsolved issue, as confirmed via both the Unity users community³ and by Unity engineers themselves⁴.

In addition, even if one succeeded in removing the interpolation from the animation, video and animation playback calculate frames to display in different way based on the current time. The video playback rounds the time up to the closest frame, while the animation playback rounds the time down to the previous frame (see Figure 3.5).

For this reason, a different solution is required for accurately synchronising camera animations with a video inside Unity.

3.1.4.4 *Solution: Back to the Basics*

To overcome this remaining synchronisation problem, we need to reject Unity tools for handling media assets, such as Video Player and automatically imported animation, and turn to a more basic approach.

First, I split both camera pose animation and camera footage into their respective samples before importing them into Unity, to enable fast random access to specific samples. In the case of animation, these samples are the camera poses associated with every video frame along with the camera's field of view, which I save to a plain text file using a 3ds Max script (Appendix A.1). The camera footage, in turn, is also split into separate video frames.

Second, I exploit that physics simulations in computer games require updating at constant time intervals (up to the current game refresh rate) and therefore have a dedicated update function. I set this time interval to match the video frame rate (e.g. to 1/25 s, i.e. to 0.04 s, to match the frame rate of 25 fps). Every time the update function is called, my script reads the current camera position and rotation from the text file and applies it to the virtual camera. At the same time, the corresponding video frame is loaded from the assets folder and displayed as a dynamic texture on the plane aligned and linked to the camera (Figure 3.6). The Unity script code can be found in Appendix A.2.

This solution corrected the image discrepancies, making all rendered frames resembling that in Figure 3.4b. The only thing to be aware of in this approach is the frame count, dependent on the media source. Some software counts frames starting from 0 (e.g. 3ds Max, media converters from video to frames), while other software starts from 1 (e.g. Foundry Nuke). Therefore, it is important to load appropriate frame numbers to ensure that the camera position and video frame correspond with each other.

³ <https://forum.unity3d.com/threads/rotation-animation-issue-with-constant-tangents.213868/> (last accessed 03/07/2019)

⁴ Private conversation (April 2017).

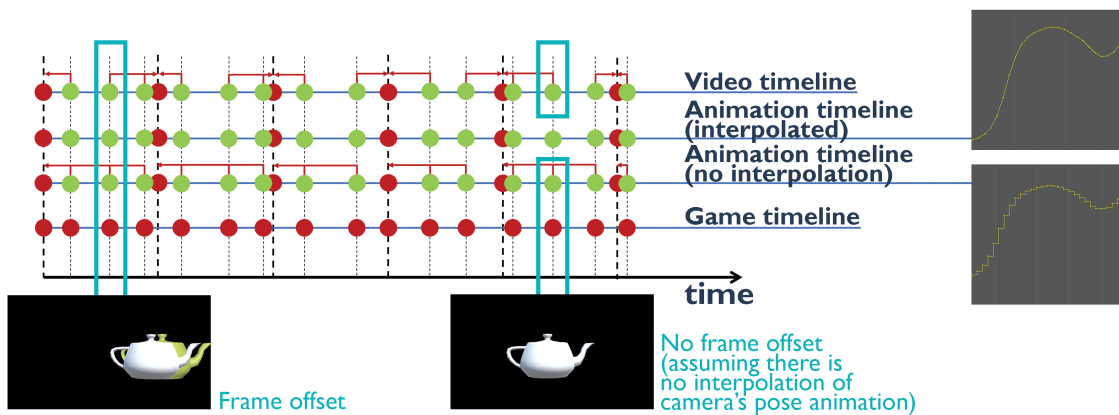


Figure 3.5: Event-driven (game timeline) and clock-driven (video and animation timeline) playbacks and the lack of synchronisation they cause. This occurs even if there is no animation interpolation, as video playback displays the frame closest to the current time, and animation playback displays the previous frame, which are not always the same.

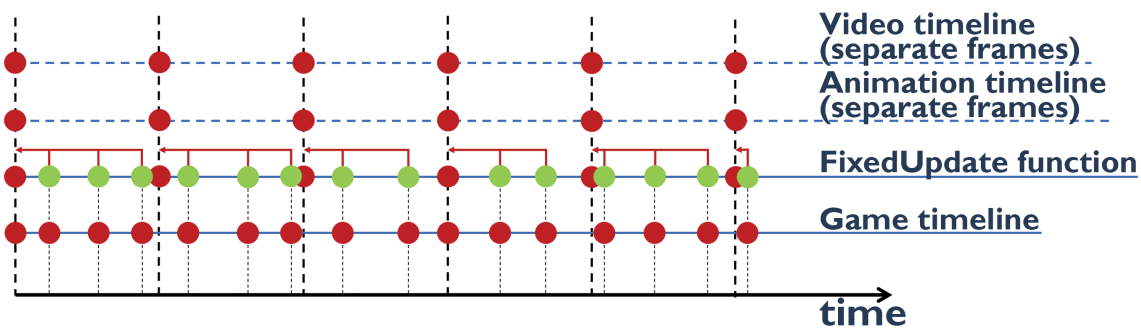


Figure 3.6: Corrected playback with video and animation split into separate frames. The FixedUpdate function, which is called in constant intervals, serves as a mapping from variable to constant frame rate and ensures that always the latest frame of both video and animation is displayed.

3.1.5 Placing Objects in the Scene

Inserting a virtual object into a real scene very often requires only a simple flat geometry, such as a plane, a table or a wall, to guide it. The output of the camera tracking contains a point cloud representing a sparse scene reconstruction, which helps in indicating an approximate place and distance from the camera to insert a CG object. Given even a noisy set of points that lie on the same flat surface, it is possible to fit a plane to these points as follows [23]:

1. Given a set of k points $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_k \in \mathbb{R}^n$, form a $k \times n$ matrix \mathbf{X} , where each row is a difference between the position vector of the point and the centroid \mathbf{c} :

$$\mathbf{c} = \frac{\mathbf{P}_1 + \mathbf{P}_2 + \dots + \mathbf{P}_k}{k}. \quad (3.10)$$

Subtracting their mean value (i.e. the centroid) from the points' coordinates protects the linear system from becoming ill-conditioned. It also later helps in aligning the plane with the geometrical centre of the points set.

2. Calculate singular value decomposition (SVD) of \mathbf{X} .
3. The right-singular vector corresponding to the smallest singular value is the normal vector of the searched plane.

I implemented this algorithm as an Unity script that takes as an input a list of GameObjects representing points, and creates a generic plane aligned with the calculated normal vector (Figure 3.7).

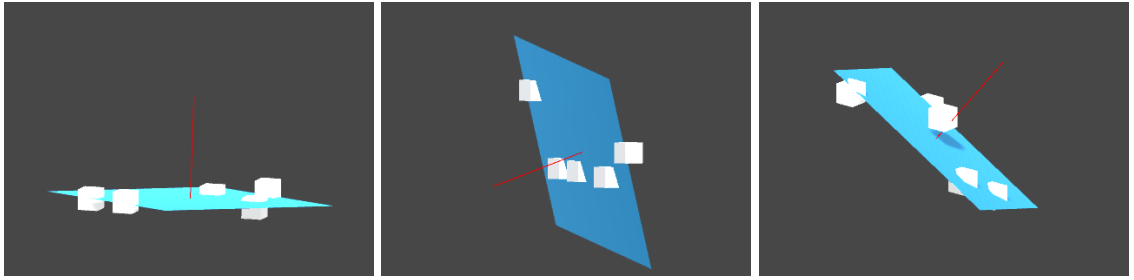


Figure 3.7: A simple illustration of the implementation of the plane fitting algorithm. White cubes represent points, while the red line indicates the normal vector of the fitted plane.

Another script aligns an object inserted into the scene with the plane and allows the user to manipulate the position and rotation of the object within the directions of the plane, using keyboard shortcuts.

Figure 3.8 depicts the plane created based on the subset of the point cloud forming the ground plane corresponding to the pavement on the image. It was scaled to show its range further from the camera, although the movement of the aligned object is not restricted to the plane boundaries, as it adopts only its directions (see Figure 3.10).

3.1.6 Image-based Lighting

I use the traditional image-based lighting approach to light the inserted objects (see Section 2.3.0.1). The objects are lit with an HDR environment map, produced from a photo of a chrome ball (Figure 3.9).

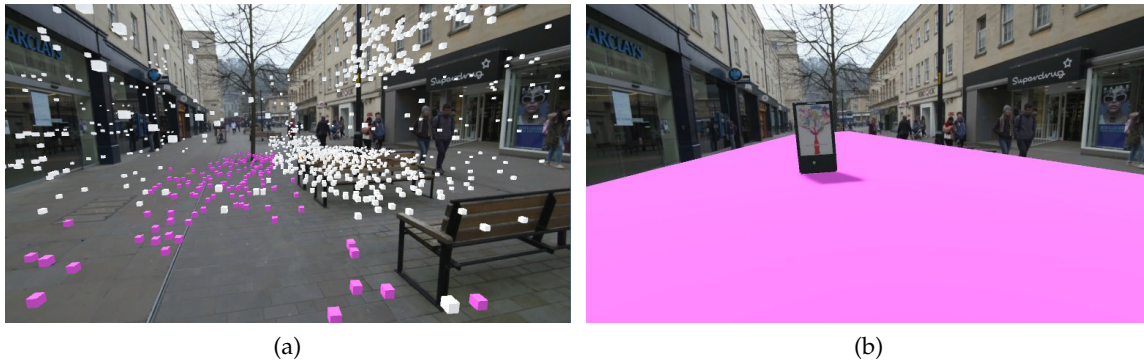


Figure 3.8: Plane fitting: (a) a subset of the reconstructed point cloud is selected as an input to the plane fitting script, (b) the plane created by the script (scaled manually to the desired size) is rendered with the 3D object aligned to it.



Figure 3.9: **Left:** HDR photos of a chrome ball, taken as close as possible to the intended place of virtual objects insertion at 90° angle to each other. **Right:** CG objects inserted into real scene, lit with an environment map extracted from the HDR photos on the left.

There are two disadvantages to this method for capturing the map. First, the photographer and the tripod are visible in the middle of the photo, covering the elements behind them. Second, half of the environment directly behind the chrome ball is heavily distorted, stretching on the ball's contour as seen from the camera. Both may result in artefacts when reconstructing the map. To compensate for this, usually, two photos of the ball are required, the second camera positioned at 90° to the first one. The two photos are then merged with special software (e.g. HDRShop [21]). This is also the approach I took.

More recently, 360° spherical cameras are often used instead of photographing a chrome ball, as they produce high distortion only on top and bottom of the image, the areas with least detail (see Section 6.1). It is also easier for the photographer to hide themselves and the tripod from the photo (usually at the bottom) or even set the timer and leave the space.

3.1.7 Differential Rendering

To look realistic, inserted CG elements must interact visually with the real scene. Casting shadows is particularly important, as objects that cast no shadows are immediately recognised as implausible, because no such objects exist in the physical world. Shadows can be introduced into a scene using Debevec's concept of differential rendering for light transport in mixed-reality scenes as explained in Section 2.3.0.2. I implemented it as a shader which combines the background frame (B) and images rendered from a set of collocated cameras with identical parameters, according to Equation 2.7 (see Figure 3.11):

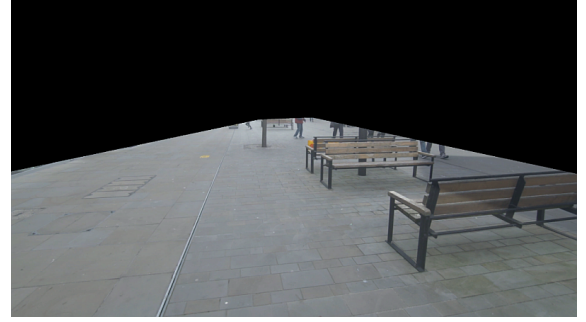
- **Camera 1:** renders only the inserted objects and the local scene (O),
- **Camera 2:** renders only inserted objects and produces their alpha mask based on the stencil buffer (α),
- **Camera 3:** renders only the local scene (L).

The shader is applied to an image plane attached to the main camera in the scene.

I assume that the CG objects are placed on a flat surface, which covers, if not all object insertion cases, at least a wide range of the most common of them. Therefore, I can model the local scene geometry with a plane created by the plane-fit script described in Section 3.1.5. The plane is based on a set of reconstructed points manually picked by the user. In his paper, Debevec estimated the reflectance model of the local scene in an iterative way. In this project, a screen space shader applied to the plane is used instead, with the footage from the camera as a dynamic texture providing the diffuse colour, which ensures the correct colour of the produced shadows. The virtual light is inserted manually by the user, to match the direction and intensity of the real light.



Figure 3.10: A 3D object (poster stand) placed at different points of the image at runtime, using the directions of the plane fitted to the sparse pavement reconstruction, and keyboard shortcuts for object manipulation.

(a) Background image (B)(b) Local scene - plane (L)

(c) Local scene on background

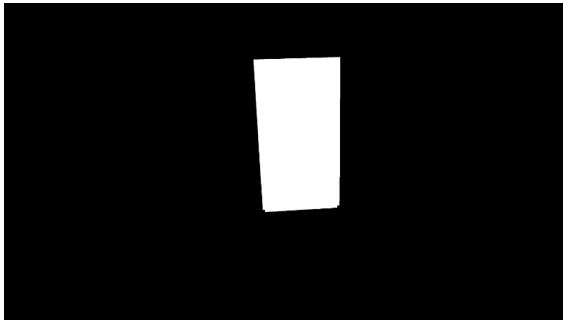
(d) Local scene with object (O)(e) Alpha mask of the object (α)(f) Final composite (C)

Figure 3.11: Components of differential rendering: (a) background image B , (b) local scene reconstruction L using a plane (view from camera 3), (c) L rendered on top of the background B , (d) local scene with objects O (view from camera 1), (e) an alpha mask based on the stencil buffer α (view from camera 2), (f) the final composite C (output from the shader, view from the main camera).

3.2 EXPERIMENTS

After the successful tests on synthetic video mentioned in Section 3.1.4, I tested my compositing pipeline on live-action footage.

3.2.1 *Camera Tracked in Postproduction.*

My test set consisted of three videos: (1) “*High street*”, (2) “*Supermarket*” and (3) “*DIY shop*”. In the first two videos, camera tracking was performed using Foundry Nuke, while Autodesk’s MatchMover was used for the last video. The advantage of MatchMover is that it produces a MaxScript code with the camera data that creates a camera with the correct position and scale directly in 3ds Max, so it can be exported to FBX format without any intermediate conversions. The disadvantage, in turn, is limited functionality and features for refining the solved camera.

Firstly, I assess the accuracy of temporal synchronisation by observing if the inserted objects stay in the same place throughout the shot, assuming that there were no errors in the input camera tracking. Any shifting in their position means that the camera pose does not match the displayed background frame. The results are satisfactory with CG objects fixed where they were placed in the scene (Figure 3.12).

Secondly, I test the interaction tools in Figure 3.13. The scenes “*High street*” and “*Supermarket*” include an implementation of differential rendering for image-based shadowing, and the scene “*Supermarket*” additionally uses an HDR map of the environment for image-based lighting. The user manually sets up the light source position, colour and intensity to obtain virtual shadows that match those cast by the real objects. All three test scenes allow the user to control the video playback as well as the position and rotation of the CG objects using keyboard shortcuts. The “*DIY shop*” scene illustrates that changing the texture of the object requires only loading the desired image file from the project assets, and can even be done at runtime, e.g. by programming a button or any other UI element to change it when pressed.



Figure 3.12: Scenes used to test the accuracy of synchronisation and compositing (from top to bottom): (1) “High street”, (2) “Supermarket” and (3) “DIY shop”. Inserted CG elements include a poster stand (1), a reflective bunny and two balls (2), and two small banners (3). Single frames extracted from the final composite sequence show that CG elements stay fixed in their places in the scene when the camera moves.



Figure 3.13: The video playback can be paused at any point and it is possible to rearrange the position of the inserted CG elements within the scene or even to change them using pre-programmed keyboard shortcuts.

3.2.2 Camera Tracked on Set (experimental)

Another type of test data was camera movement recorded on set with the use of experimental hardware setup. The footage was recorded by the Google Tango tablet camera, while additional sensors with which this device is equipped allowed to estimate the position of the tablet in six degrees of freedom while recording [36].

However, it is difficult to assess the accuracy of the output compositing, due to some unsolved issues with the recording pipeline. The Tango tablet was designed for augmented reality applications, and therefore provides a good synchronisation between the estimated pose and a camera view in real time, but it lacks API functions for recording the video. A mobile application which I built for that purpose is still in its experimental phase, and therefore, the recorded video suffers from dropped or doubled (or even tripled) frames, which makes some parts of the output compositing seem visually not synchronised, while the general path of the virtual camera matches the movement of the real one (see Figure 3.14).

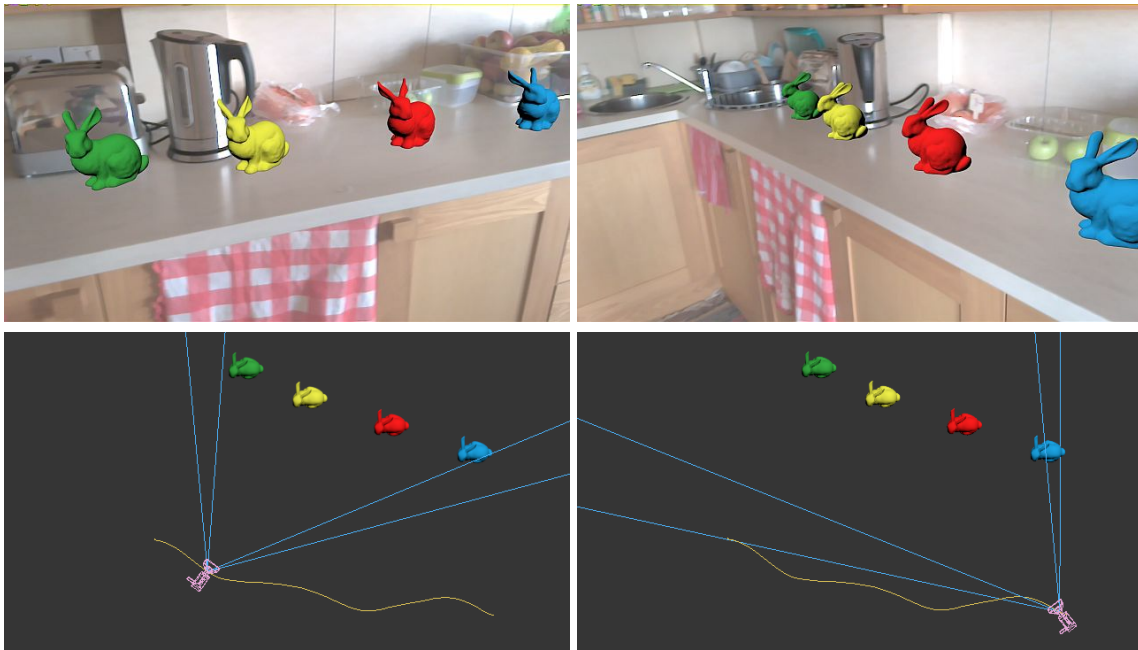


Figure 3.14: Frames of the output compositing, where all camera pose data was recorded on set with a Google Tango tablet. The background footage was recorded using the built-in tablet camera.

3.3 DISCUSSION

I presented a general pipeline for dynamic mixed-reality compositing in Unity. My pipeline loads camera pose animation and field of view from a text file into a game engine, and it is therefore not limited to a particular media file format or the source of the animation, providing that the input data follow the file template. I used camera data obtained from different sources (3D camera tracking software, motion tracking hardware) without any changes in the compositing pipeline. This information is then applied to a virtual camera used to render 3D objects matching the camera footage displayed in the background. An important benefit of using a game engine environment is that it already provides a variety of tools to interact with objects in real time, for example to change their position, orientation or texture, or to replace them with other objects.

There are a few limitations to my approach. The main limitation is a necessity to split the video footage from the camera into separate frames. This increases project size and thus affects its loading time, which may be problematic in on-line or mobile application scenarios. However, this is the standard approach for handling camera footage in VFX pipelines and is performed regardless. The quality of the compositing also relies heavily on the accuracy of the initial camera tracking. However, any software can be used in principle, which makes our approach more flexible to use for artists. As with any matchmoving, artists need to carefully choose features to track. In particular, the output point cloud should have plenty of points around the spot where the CG elements are intended to be inserted. That is because the plane fitting script works best with large surfaces containing many points. However, scene reconstruction is sparse, and it is sometimes difficult for the user to select the sufficient subset of points. It is also usually very noisy, making it challenging to indicate which points lie on the same surface. Lastly, the rendering quality of a game engine cannot be compared with the production quality achieved by specialised offline production rendering software, although game engines are constantly improving their rendering quality, and artists started considering them as a tool mature enough to use them to make animations^{5,6}. However, the existing rendering quality is sufficient for previsualisation and market research purposes.

The current scene lighting is adjusted manually, and getting satisfactory results may require a significant amount of time for refining the position, intensity and colour of the main light source (i.e. the sun). This could be automated using automatic light estimation based on HDR environment maps. I also assumed that the local scene can be approximated by a plane. It would be worth considering more complicated shapes, and methods of their acquisition, to create a more precise model of the close vicinity of the inserted objects. That would allow them to become occluded by the real objects and thereby increase the number

⁵ <https://cgsociety.org/news/article/4527/creating-a-short-film-in-ue4-with-darko-subotin> (last accessed 05/07/2019)

⁶ <https://www.awn.com/animationworld/unity-releases-cg-short-sherman-showcase-real-time-production-template> (last accessed 08/07/2019)

of their possible locations within the scene. However, planes work well in many practical scenarios.

My compositing approach can easily be extended to panoramic 360° videos, by adopting a 360° video pipeline for camera tracking, which I described in Chapter 5. Additionally, a 360° input video could be directly used as a dynamic environment map. Using inverse tone mapping, the video can be converted from low dynamic range to pseudo high dynamic range (HDR), as proposed by Iorns and Rhee [46] and Rhee et al. [80]. Chapter 6 contains a complete dynamic mixed-reality compositing pipeline for 360° videos.

USE CASE: PETROL STATION RESEARCH STUDY

I tested my dynamic mixed-reality compositing pipeline on a real-life example. It was motivated by the past research study conducted by Checkmate VR for a fuel brand. In the study, they used a fully computer-generated environment to simulate the customer journey to a petrol station (see Figure 4.1). This is how the company explains the study and difficulties associated with delivering multiple versions of the video and working across different time zones:

Checkmate VR recently completed a research project for a fuel brand, we needed to generate 100 different combinations of a journey a person might do when filling up their car, in each journey different POS [Points of Sale – author’s note] and designs will be used. This study was done for UK and German forecourts and included 3 different branded stores per country.

To break down the journey we needed to produce:

- *A video of the drive into the forecourt and pulling up at the pump,*
- *An interactive photosphere [360° image – author’s note] for the participant to choose their fuel,*
- *A video of the walk from the pump to the store counter,*



Figure 4.1: The CG environment of a petrol station and its neighbourhood created by Checkmate VR for their online survey. Image courtesy of Checkmate VR/Dc-activ.

- *An interactive photosphere of the counter where sweets and chose of cash or card could be selected,*
- *Finally, a video of the walk back to the car.*

Each scenario had 14 different zones that could change, and each zone had up to 8 different items, such as different POS, and pump designs. In total, we needed to create 300 videos and 200 photospheres per country. It took on average 12 hours to complete the rendering per country. Every time a POS element changed or we spotted mistakes we needed to re-render.

The feedback loop was very long with this technique, we would complete the changes requested and set the scene off to render. As it would take 2 machines 12 hours to render everything this was usually done overnight. The next morning we would then QA the videos to ensure nothing had gone wrong, on occasion it had, so therefore we delayed sending to the client. As the client was in the US they wouldn't start looking at the videos until around 3pm our time, if they spotted any problems or wanted to change something it would normally have to wait until the next day for us to fix and set the renders off again. Once the client was satisfied it would then go to their client for review. The whole process to client sign off could take 4 - 5 days due to time zones and render times.

Having a "real-time insertion" player would have reduced our time dramatically as we would only need to render out 9 videos per country (1 for each part of the journey and brand).

At Checkmate VR, we focused on one part of the customers journey, which was a video of the drive into the petrol station's forecourt and pulling up at the pump. We created a similar example of the drive, but with a mixed-reality video. Having in mind their prospective clients, the company intended to use it as a proof of concept, and to add it to their portfolio. We recorded real footage of a car drive, and applied the dynamic mixed-reality compositing pipeline to insert or modify elements of the video. The final scene was also adjusted by the 3D artists from Checkmate VR in terms of lighting and positioning of the virtual objects to make the insertions as realistic as possible. The output is a Unity project with various elements, such as a bus stop, a poster, a pump, a totem or a blimp inserted into the video of a drive to a petrol station (see Figures 4.2 and 4.3). The flat advertisements (e.g. posters and the billboard) come in two versions and can be changed in real time. All insertions can be switched off and on, also in real time.

4.1 USER STUDY

After building the proof-of-concept scene, the next step was to run a user study to compare the experience of participants in the fully computer-generated and the mixed-reality environments and to test how useful the mixed-reality environment is for conducting

market research studies. Unfortunately, there were some restrictions to the mixed-reality video asset, which were known at the time of designing the study, yet could not be avoided. This occurred because the idea to run the study emerged after the mixed-reality video had been created as a proof of concept. Therefore, the mixed-reality environment is not identical to the one in the computer-generated video used in the previous market research. The petrol station and its surroundings modelled in the computer-generated environment is a generic one, so there is no real-life equivalent that could be recorded for the mixed-reality version. Also, the background video was recorded with no prior experience in such recordings, so the camera was located in the most convenient place, namely on the passenger side, where it could not disturb the driver. Recording video from the passenger's perspective carries a risk of interfering with study participants' perception of the environment.

Nonetheless, even in such a form, the study produced some significant results and led to valuable conclusions, also beyond the scope of the original hypotheses. Below I describe the study in more details, including the method and the analysis of the results.

The study was designed jointly with Tim Jarvis and Michael Hagerty from Checkmate VR, and consulted with Dr Christof Lutteroth from the University of Bath.

4.1.1 *Method*

4.1.1.1 *Design*

This research study was motivated by an interest in a degree of realism people associate with fully computer-generated environments vs. mixed-reality environments (called CG and MR, in short, in the rest of this chapter). If the associated degree of realism turns out to be similar for both, or higher for MR, it would confirm that MR environments, which are quicker to build and render, are likely to replace the fully CG ones as a market research tool. Also, if they encourage more natural behaviour and, in consequence, lead to more natural decisions, they are an even more desirable asset.

I formulated the following hypotheses:

H1: People prefer MR environments as they are more realistic.

H2: People behave more naturally in MR environments than they do in purely CG ones or, at least, they behave in the same way in both.

H3: MR environments have all the functionalities of fully CG environments and, therefore, can replace them as a market research tool.



(a) original video



(b) highlighted areas of insertions



(c) first version of the augmented video



(d) second version of the augmented video

Figure 4.2: Mixed reality version of the drive to the petrol station (close-up): (a) an original video frame, (b) highlighted areas indicate where the augmentations will be inserted, (c,d) the first and the second version of the final compositing. Transitions between these four states can be done in real-time during playback. *Image courtesy of Checkmate VR/Dc-activ.*



Figure 4.3: A petrol station forecourt with inserted virtual elements: a poster, a pump, a totem, and a billboard and a blimp in the background. One of the two augmented versions of the video. *Image courtesy of Checkmate VR/Dc-activ.*

The first two hypotheses were motivated by the work of Slater et al. [88] who proved that greater visual realism, leads to more realistic behavioural responses in a virtual environment.

The research study was designed as an online questionnaire with videos, similar to the surveys usually designed by Checkmate VR and Dc-activ for their clients. The study was conducted in two stages: first a pilot study and then the main study.

4.1.1.2 Participants

The user study contained the videos of a drive to a petrol station and required the participants to imagine they were driving a car. Therefore, all recruited participants were adult active drivers, where active meant that they drove a car and filled it up with petrol less than six months before.

Originally, Checkmate VR intended to recruit 200 participants, as they expected very subtle differences in perceptivity between the two videos. In anticipation of the difficulty of this task, as an incentive, every participant was entered into a prize draw of two £50 Amazon vouchers. The final number of participants was, unfortunately, much lower than expected, although still significant to conduct a valid study.

In the pilot study, there were 10 participants, 8 male and 2 female, 23 to 53 years old. In the main study, I got 41 responses. There were 26 male and 14 female participants (one person preferred not to reveal their gender), aged 20 to 58. They were recruited from different backgrounds: from University of Bath staff and postgraduate students, from employees of startup companies in Bristol via their email list, and amongst friends via individual messages and Facebook.

4.1.1.3 *Materials*

The study was conducted through an online questionnaire. It consisted of six parts (they all can be found in Appendix B):

- I. A participant information sheet and consent form (Section B.1).
- II. Qualifying questions asking participants about the estimated date of their last visit to a petrol station and filling up the car to confirm that they are eligible for the study (Section B.2).
- III. Introductory demographic questions, and a few questions to make the participants think about their last visit to a petrol station (Section B.3).
- IV. Two videos of a drive to a petrol station, one rendered from a fully CG environment (Figure 4.1), and the second one the mixed-reality one (Figures 4.2d and 4.3). They were used in the perceptivity test to check how many elements participants remember from each video (Sections B.4 to B.7).
- V. The Slater, Usoh, and Steed (SUS) presence questionnaire (which proved to outperform the popular Witmer and Singer questionnaire [114] when applied to real environments [90, 110]). It consists of six questions, which I modified to fit in the study context in a similar way as it was done by Usoh et al. [110] (Section B.8):
 - 1 *Please rate your sense of being in the car driving to a petrol station, on a scale of 1 to 7, where 7 represents your normal experience of being in a place.*
I had a sense of “being there” in the car driving to a petrol station:
1. Not at all ... 7. Very much
 - 2 *To what extent were there times during the experience when the video was the reality for you?*
There were times during the experience when the petrol station was the reality for me...
1. At no time ... 7. Almost all the time
 - 3 *When you think back to the experience, do you think of the video more as images that you saw or more as somewhere that you visited?*
The petrol station seems to me to be more like...
1. Images that I saw ... 7. Somewhere that I visited

- 4 *During the time of the experience, which was the strongest on the whole, your sense of being in the car driving to a petrol station or of being elsewhere?*

I had a stronger sense of...

1. Being elsewhere . . . 7. Being in the car

- 5 *Consider your memory of watching the video. How similar in terms of the structure of the memory is this to the structure of the memory of other places you have been today? By “structure of the memory” consider things like the extent to which you have a visual memory of the video, whether that memory is in colour, the extent to which the memory seems vivid or realistic, its size, location in your imagination, the extent to which it is panoramic in your imagination, and other such structural elements.*

I think of the petrol station as a place in a way similar to other places that I’ve been today...

1. Not at all . . . 7. Very much so

- 6 *During the time of your experience, did you often think to yourself that you were actually in the car driving to a petrol station?*

During the experience I often thought that I was really driving a car...

1. Not at all . . . 7. Very much so

VI. General preference questions (Section B.9).

4.1.1.4 Procedure

At the beginning, participants were asked to read the participant information sheet and sign the consent form by ticking the box at the end of the form (Part I). Then, they were supposed to answer qualifying questions (Part II). Only after signing the consent form and meeting the qualifying criteria they could proceed to the next parts of the questionnaire.

In the pilot study, the participants were seated individually in a quiet place in front of a computer, and supervised in an unobtrusive way. The two versions of the questionnaire were manually assigned to them, interchangeably based on the order they had signed up. During completing the study the participants could ask questions and get any unclear task explained.

In the main study, participants completed the study online at their leisure. The surveys were randomly assigned to a participant by a java script that chooses one of the two versions and redirects to it. In Part IV, 21 participants watched the CG video first and the other 20 watched the MR video first.

4.1.2 Analysis

4.1.2.1 Perceptivity

For every question where the participant was required to remember elements from the videos, they scored 1 point for a correct answer. For a wrong answer, 1 point was deducted from their score, and for no answer in multiple-choice questions they gained no points. A different scoring applies to the MR video in the multiple-choice question, due to the number of elements from the list present in both videos. There are six elements in the CG video but twelve in the MR video. Therefore, the perceptivity score for the latter requires normalisation.

I applied two normalisation factors to the score. The first factor is a natural choice of the ratio between the number of elements from the list present in both videos. As there are twice as many elements in the MR video, the correct answer scores 0.5 points and, respectively, for the wrong answer 0.5 points are deducted from the score. However, that assumes that the participants' ability to recall elements scales linearly with the number of elements. This is not always true. Therefore, I also applied a different normalisation factor to the score, based on the ratio between the average number of elements noticed by participants in the CG video and the MR video. In the analysis, I included the resultant scores calculated with both normalisation methods.

The results for single-selection questions are presented in Table 4.1. The majority of the participants could recall the brand of the petrol station from the video (81% in the case of the CG version and 88% in the case of the MR version), but, in general, they did not remember the price of the petrol. Only 12% and 20%, respectively, made an attempt to recall it, and only 8% of participants in both cases were correct or close to the actual price. It is worth noting that there were only two petrol prices in the CG video and that the image was sharp, while in the MR video there were four petrol prices, and a motion blur (one participant commented on the blurriness of the image prevented them from clearly seeing the price). Therefore, it is counterintuitive that more participants claimed to remember the price in the MR video if it was more difficult to spot.

The questions about the size of the petrol station and the side of the road the car was driving on were answered correctly by most participants. A slightly lower score for the question about the side of the road in the MR video may be caused by the camera having been placed on the passenger's seat, and not on the driver's side of the car. This high score indicates that both types of videos provide the viewers with a good sense of space.

However, when asked about the number of other cars and buildings, most of the participants managed to answer correctly only in the case of the CG video and the number of cars. It is understandable, as there are no cars in that video and the general emptiness of the scene makes it easier to notice. The poor performance of participants in the rest of these questions indicates that they paid little attention to their surroundings, being too

Table 4.1: Percentage of correct answers to single-choice questions in the perceptivity test.

	CG	MR
Petrol station brand	81%	88%
Petrol price (attempted/correct or close enough)	12%/8%	20%/8%
Size of the petrol station	96%	100%
Side of the road	96%	92%
Number of other cars	96%	48%
Number of other buildings	46%	36%

focused on the driving and the petrol station. Therefore, if there are any market research stimuli placed outside the petrol station, they might be missed.

In the multiple-choice question, where participants were asked to tick off elements on the list they remembered from the video, they performed similarly for both videos if the normalisation factor of 0.5 was used for the MR video score. However, the MR video obtained a higher score than the CG video when I applied the second normalisation factor. It was calculated based on the average number of elements recalled in both videos, which were 3 and 4.56, respectively, for the MR and the CG video. This gives the normalisation factor of 0.66. Nevertheless, the overall perceptivity score, which comprises both single-choice and multiple-choice questions, was in favour of the CG video regardless of the normalisation factor used.

Table 4.2 contains the average perceptivity scores individually for single-choice and multiple-choice questions, and the overall score for the whole section.

Additionally, 9 participants out of 25 (36%) did not notice any alterations or insertions to the MR video, even the obviously out-of-place blimp in the sky. One person claimed to notice just one such element, a pothole, which in fact was a part of the original footage. Interestingly, it seems that some participants picked the objects they *associated* with the environment and which they *thought* should be featured in the video, not the ones they actually *remembered seeing* (such as a fire extinguisher or a zebra crossing).

Table 4.2: The average perceptivity scores in the single-choice and multiple-choice questions, and combined for the whole perceptivity section. Due to different numbers of elements to spot in the CG and the MR video, for the latter, two normalisation factors were taken into account. First is based on the ratio between the numbers of elements to notice in both videos. Second scales the score according to the ratio between the average numbers of elements noticed by the participants in both videos.

	CG	MR norm. factor 0.5	MR norm. factor 0.66
Perceptivity score (single-choice questions)	2.46±1.63	1.52±2.02	1.52±2.02
Perceptivity score (multiple-choice question)	1.85±0.78	1.82±0.75	2.40±0.99
Overall	4.31±1.93	3.34±2.06	3.92±2.12

4.1.2.2 *Feeling of Presence*

Due to unclear labelling in the pilot study, only participants from the main study (42 people) were considered in this part.

Table 4.3 shows the scores from the SUS questionnaire for each question. The overall score of the MR video was higher than the score of the CG video.

However, the order in which the videos were watched mattered. When the CG video was watched first, it scored higher in Q4 and Q6 than the MR video. When watched second, i.e. participants could compare both videos, the CG one scored much lower than the MR one. Also, the CG video watched second scored much lower than the same video watched first. And vice versa, the MR video scored higher in all questions than the CG video when watched second, and also much higher than it scored when watched first.

4.1.2.3 *General Preference*

When asked which of the two videos they preferred, 76% of participants chose the mixed-reality version, while only 22% picked the CG video. One person (2%) did not have any preferences. Similarly as for the feeling of presence, the order in which the videos were watched also mattered. The participants were less likely to choose the CG video if they watched the MR video first (Figure 4.4).

The majority of those who commented on preferring the MR video mentioned realism as a factor that influenced their choice (24 people). In some cases, it was combined with a feeling of immersion and with a difficulty in distinguishing real and CG elements. The other factors in favour of MR included more natural car movement (6 people), more details in the scene (4 people), the petrol station resembling one visited in real life (2 people), and

Table 4.3: The scores (means and standard deviations on a scale from 1 to 7) of questions 1–6 in the SUS questionnaire for the CG and the MR videos. The overall scores are followed by individual scores for each video when viewed first and second. The last two rows show the change in score for each video, dependent on the order of viewing.

	Q1	Q2	Q3	Q4	Q5	Q6
CG overall	3.46±1.72	2.85±1.54	3.05±1.80	3.49±1.79	2.73±1.43	2.44±1.69
MR overall	5.07±1.40	4.62±1.71	4.95±2.00	4.67±1.75	4.38±1.79	3.43±1.93
CG when watched first	4.38±1.50	3.62±1.53	3.86±1.74	4.33±1.59	3.20±1.33	2.95±1.91
MR when watched first	4.70±1.66	4.35±1.87	4.20±2.26	4.00±2.00	4.00±2.08	2.65±1.69
CG when watched second	2.50±1.40	2.05±1.10	2.20±1.47	2.60±1.57	2.25±1.41	1.90±1.25
MR when watched second	5.38±1.07	4.81±1.57	5.62±1.50	5.29±1.27	4.67±1.46	4.05±1.88
Change in ratings between the two (CG)	-1.88±2.05	-1.57±1.89	-1.66±2.28	-1.73±2.24	-0.94±1.94	-1.05±2.28
Change in ratings between the two (MR)	0.68±1.97	0.46±2.44	1.42±2.71	1.29±2.37	0.67±2.54	1.40±2.53

more action in the streets (1 person). One person also chose the MR version because it was more rural, but this reason seems to be too specific to individual preferences.

The participants who preferred the CG video, in turn, mentioned its perfect, or even “utopian” look as a positive feature (2 people). They also liked the simplicity and clarity of the video with fewer details to distract attention (3 people). One person additionally admitted that it allowed them to remember more from the video. Two participants liked the smooth and calm car movement, and one person mentioned a more recognisable petrol brand. The other two comments seem to be of no value to this study, as the participants mentioned that “it was cool to see a CGI model of a petrol station”, and that “it was more *different* to my daily experience of buying petrol”.

On a few occasions, the participants justified their choice by commenting on what they did not like in the other video. In the case of the MR video, they most often complained about feeling like a passenger, not the driver, and that the petrol station brand was not a UK brand with prices not in pound sterling. The CG video, in turn, felt “fake” and “like a video game” with unrealistically smooth camera movement (“it felt like the car was on train tracks”). In addition, the scene background was criticised for flat-looking buildings and disproportionally big trees.

In the second question, the participants were asked to choose in which of the two environments they were more likely to behave like in reality. The question has different forms in the pilot and the main study and, therefore, I separate their analysis.

In the pilot study, it was an open question. 6 out of 10 participants chose the MR video, while only 2 picked the CG one. One person declared that there was no difference in their behaviour between the two videos, and another person stated that if it was not possible to

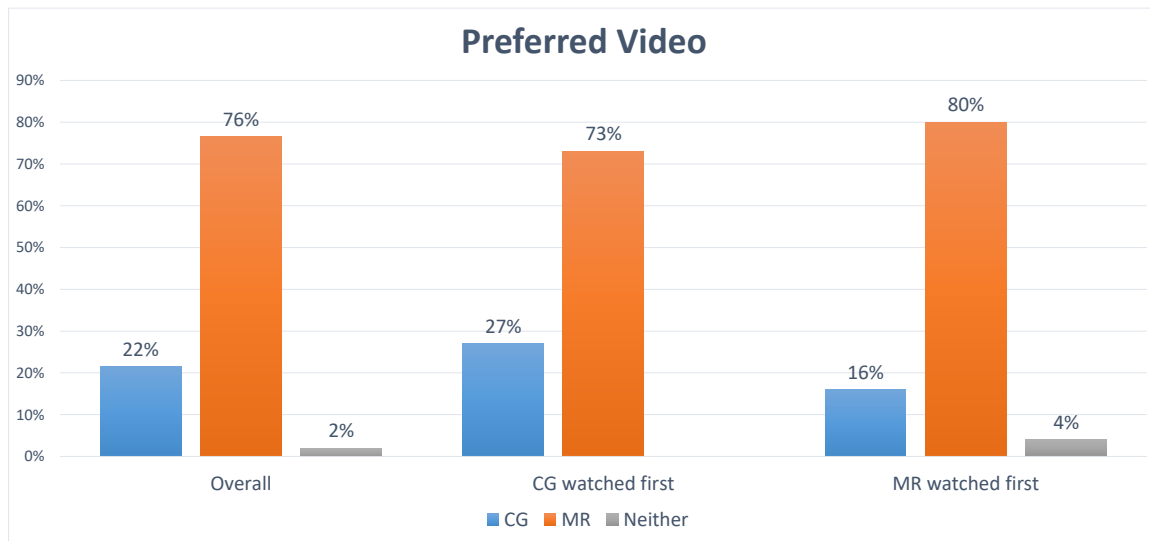


Figure 4.4: The percentage of participants who preferred the CG video, the MR video or neither of the two. The overall result combines the two versions of the survey, while the other two results illustrate the preferences depending on the order of watching.

control the car movement, neither video would make them behave naturally. The answers do not seem to be dependent on the order of watching the videos but perhaps that was caused by a small study sample.

In the main study, the question had been changed to single-choice. Having just two options to choose, 39 out of 41 participants chose the MR version as the environment in which they are more likely to behave like in reality. That includes all participants who watched it as the first video.

The answers to the third question, which asked about anything weird spotted in the video, by some means extend the answers to the first question in this section. In the first question, the participants indicated the preferred video. In the third question, they gave more insight into the reasons for their choices. Mostly, they come down to pointing out the imperfections of both videos, especially the CG one, which also leads to better understanding why the participants preferred one video over the other.

In their answers, 13 people complained about the smooth, slow and unrealistic car movement, and a pivot turn in the CG video. Another 7 people noticed the lack of other cars and people, two of them in both videos (which is unexpected for the MR video, where there were plenty of other cars on the road and some in the petrol station). The third most commonly criticised element of the CG video (5 people) was a disproportion in size between the background trees and buildings and the rest of the scene, which caused an incorrect parallax motion when the car was moving. Other single comments mentioned a fixed camera view in both videos (inability to move head while driving), lack of signalling in the CG video, and bounciness of the car movement in the MR video.

It is surprising that only 9 people noticed and considered weird a blimp in the sky (one person was unsure if this counted as “out of place or just because they are not usually seen where [they] live”). Also, only 4 people guessed that the MR video included some CG elements as well as real ones. 11 people did not notice anything unusual in both videos.

4.1.3 Discussion

The results, and especially the answers and comments to questions in Part VI (General preference), confirm H1. Study participants preferred the MR video, and the realism was often mentioned as a reason.

The study also confirms H2, as people admit in Part VI that they would behave more naturally when viewing the MR video than the CG version of the same scene. However, even though the results of the Part V (Feeling of presence) are also definitely in favour of the MR video, the overall presence score (and individual scores for each question) are not very high. The reason is that, above all, the participants were watching only a video without sound, and they could not control the car or look around. Therefore, videos cannot provide a complete feeling of presence, which is an essential element of realism. Consequently, in the context of this study, we can only talk about relative realism, not the absolute one.

The main functionality of CG video assets in market research, which I considered in H3, is their flexibility. CG environments can contain any required object in any form, also in many different versions. This study proved that the CG objects composited into an MR video do not break the realism of the scene, and often are not even noticed as out-of-place. That gives the MR environments the same flexibility as their CG equivalents. However, when we compare the amount of information the participants can recall from both types of videos, there is a definite advantage to use CG videos in this case, which makes H3 just partially confirmed.

On the other hand, this study also revealed that people notice more in the CG video because there is a limited number of options and the environment is simplified. That does not resemble real-life situations. Therefore, in view of the obtained results, and especially participants’ comments, the assumptions on which the original hypotheses were based may require verification. The question arises if the goal to create MR assets is to replicate all the functionalities of the existing CG assets or to make the assets more realistic, with all associated flaws, for better reliability of market research studies.

In this particular use case, we may base on the output of the research study to formulate recommendations on how to improve current video assets, both fully CG and MR. Emulating bumpy car movement and natural turning, remodelling the background to correct scale and adding other cars and people to the CG scene would soften its artificial feel. It should be noted that it would require a significant amount of CG artist work. In the case of MR videos, while compositing meets the requirements, more care should be put into

recording the background footage, and the choice of the inserted content. Both videos would benefit from adding sound.

4.1.3.1 *Recommendations for An Improved Future User Study*

I am aware of the limitations of this study due to factors out of control. In particular, the environment in the MR video is not identical to the environment shown in the CG video, which makes the comparison more difficult and less reliable. This is because the MR video was originally created as a proof of concept rather than an asset for a particular study, and therefore did not intend to replicate the previous CG version of the drive to a petrol station. There were also some issues with the video itself, not associated with mixed reality, such as a camera placed on the passenger side, which for some people breaks the feeling of driving the car. Finding all these imperfections was beneficial for the project, as it will help to improve the video in the future.

Based on experience gathered through this user study, I compiled a list of recommendations and design ideas that might be useful for conducting an improved version of this or a similar study in the future. Provided that the necessary financial resources can be secured:

1. The CG environment should be modelled based on a real place, and the same place should also be recorded as a background for the MR video.
2. I do not recommend reusing existing environments as assets. Both CG and MR environments should be purpose-built, with no details left to chance.
3. The CG environment should be also modelled with more details to minimise the bias in perceptivity, which favours simplified environments with less elements.
4. Place the same items, and the same number of items in both scenes if participants will be asked to recall them. This eliminates the necessity to scale the answers.
5. Increase the level of immersion by adding sound to videos. The increased immersion leads to the increased presence, and that, in turn, to more realistic responses [88].

We can go even further and exploit all the possibilities created by the game engine environment. It provides us with programmable controllers that can be linked to the car movement. These features, combined with a programmable moving chair similar to those in 5D cinemas, could be used to build a simulator-like experience. This setup ensures the maximum of immersion for the CG and the MR videos alike, and I expect it to result in the most realistic response possible in both cases.

It would be interesting then to measure the difference in the obtained perceptivity score between the two videos as well as the presence score. I presume that the latter would increase in both cases, with the MR video still obtaining the higher score as the visually more realistic of the two environments. Nevertheless, the difference in score between the two videos might be smaller than in the current results.

4.1.3.2 *Conclusion*

At present, videos are still the most common type of asset for market research user studies, as it is expensive and cumbersome to conduct a study with hundreds of participants using VR headsets. However, as this study shows, even real video does not make the scene feel absolutely realistic. In my opinion, the future of market research lies in truly immersive environments, which become cheaper and cheaper and more widely accessible.

In the next chapter, I focus on omnidirectional videos, which are characterised with both immersion and realism. Combined with mixed-reality compositing, they are a natural candidate for assets in next-generation market research studies.

OMNIDIRECTIONAL STRUCTURE FROM MOTION

Camera tracking is one of the three main technical challenges present in the task of inserting computer-generated elements into live footage, which I described in Section 1.2. It is well-understood for standard perspective cameras, either as an off-line process using structure from motion (SfM) or real-time visual simultaneous localisation and mapping (SLAM) [66]. However, the recent introduction of 360° cameras into the creative industry required redefinition of the standard approaches to camera tracking to adjust them to a different camera geometry. This is still an ongoing process and a subject to research.

In my work, I focus on inserting CG elements into pre-recorded video footage rather than on real-time applications, and therefore, I adapted structure from motion in my pipeline.

Despite the unquestionably different geometry of an omnidirectional camera compared to the pinhole camera model (see Section 2.4), the same epipolar constraints principles apply to it. This can be exploited to formulate the foundation of an omnidirectional version of structure from motion algorithm (see Section 2.5.1).

Structure-from-motion algorithms assume known image correspondences. Some early work used synthetic points or correspondences obtained from manual annotations, but practical methods include feature detectors and descriptors and feature tracking (see Section 2.5.2). Feature detection and matching can be expensive for longer videos and perform poorly for scenes with repetitive elements. Therefore, I track features over multiple frames to speed up execution and reduce mismatched repetitive features.

There are a couple of different approaches to feature tracking for 360° videos. Either, tracking is performed on image components (fisheye or perspective) before stitching, or the already stitched image is remapped to the cubemap format and considered as separate videos from a virtual six-cameras rig (see Section 2.5.3). I track features directly on equirectangular images, which seems to be approached warily by the research community, and no known publication ventures to do that. In my work, I proved that (at least for the most common horizontal camera movement), equirectangular images are reliable enough, despite their high distortion, to be used without any preprocessing in tracking.

Challenges exist throughout: tracking 360° camera motion requires robust structure from motion, with many current methods reconstructing individual views from the set

of two, six, or even more input camera views. To create a single equirectangular image from the separate views, stitching algorithms tend to distort the boundary pixels between the input images for visually seamless stitching (Figure 5.1). This occurs not only in consumer cameras but also in professional ones. Distorted boundary pixels lead to incorrect re-mapping the feature position on the sphere and in consequence to tracking errors. Therefore, it is justified to use a set of images before stitching for tracking to avoid stitching errors, even if that complicates the whole process. However, most consumer 360° video cameras produce an *already-stitched* 360° video in equirectangular format, which precludes existing many-input-view 360° structure from motion techniques. Re-projection of this equirectangular video back onto the separate views transfer the stitching errors.

Nonetheless, with a large number of tracking features evenly distributed across the image, stitching errors do not contribute significantly to the final solution. Besides, if the same stitched video is to be used as a background for compositing, it is desirable to obtain the solution that matches the background, even if not necessarily accurate to the original camera parameters and scene geometry. Therefore, feature tracking directly in equirectangular format has its advantages.



Figure 5.1: Examples of stitching artifacts in 360° spherical images. **Left and right:** duplicated objects (“ghosts”), **middle:** broken horizontal lines.

5.1 OMNISFM PIPELINE

After analysing related work for structure from motion for omnidirectional (not necessarily spherical) and perspective cameras, I decided to implement a custom 360° structure-from-motion pipeline (OmniSfM). The motivation was to select the simplest and easiest to implement steps from different algorithms and, in result, to contribute with a simplified and reliable version of the SfM algorithm for spherical cameras. My implementation consists of three main modules:

- **Feature Tracker:** I use a modified KLT tracker [63, 104] which finds point correspondences between equirectangular video frames (Section 5.1.1). There is no prior art for working directly in the equirectangular domain. Also, to maximise its tracking accuracy, I based my implementation on my experience with commercial tracking software for perspective cameras. I extended my implementation with several tunable parameters adapted from the off-the-shelf perspective camera tracking, where they improve the tracking flexibility.
- **Camera Solver:** I apply the epipolar constraint for spherical cameras to calculate relative camera poses and triangulate the 3D positions of the tracked points (Sections 5.1.2 and 5.1.3). In particular, I substitute the catadioptric camera model in the solution proposed by Chang & Hebert [13] with the spherical camera model.
- **Bundle Adjustment:** Instead of a standard one-step bundle adjustment, I introduced the two-step hierarchical bundle adjustment to refine both the initial camera poses and reconstructed points (Section 5.1.4). The two-step approach increases the accuracy of the final solution and speeds up the convergence. I also use the tangential reprojection error as the cost function, which is faster to compute than the angular distance usually used for measuring the error on a sphere and produces a smaller residual error.

I describe all these steps in more detail below.

5.1.1 Feature Tracking

My approach takes as input a sequence of equirectangular images and performs a modified version of KLT tracking to obtain point correspondences between consecutive frames (Figure 5.2). Sequential tracking of video sequences produces fewer outliers and mismatched points, and is significantly faster than all-pairs feature detection, description, and matching when applied to video sequences of hundreds of frames or more, as in my case.

My implementation is adapted from the Accord.NET Extensions KLT tracker [48]. I make four changes:

1. To accommodate the geometry of equirectangular projection and extend track lengths, I wrap feature point locations along the x-axis such that tracks do not stop at left and right image edges.
2. To improve robustness and reliability with long video sequences, I switch from frame-to-frame tracking to template-based tracking. This extracts and stores the template from the first frame in which a feature appears, which reduces sliding of the tracked point.
3. To create longer feature trajectories, particularly for features created towards the end of the video, I follow a forward pass of feature tracking with a backward pass which seeks to extend existing feature trajectories back in time.
4. I compute bidirectional tracking error to detect inconsistent tracks [49]. If it exceeds two pixels then the feature trajectory is terminated. I do this from one frame to the next rather than evaluate the whole trajectory. That allows me to eliminate the feature from the frame where it drifted away, but to keep the valid part of its trajectory.

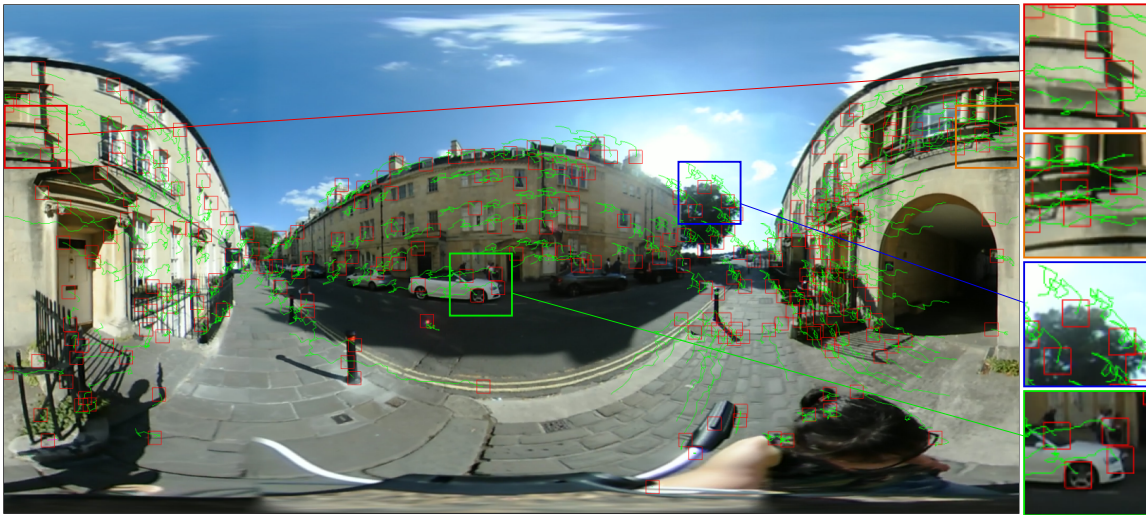


Figure 5.2: Omnidirectional feature tracking in progress. Note that the red crop (on the left edge of the equirectangular image) contains feature trajectories (green lines) that have been tracked to the orange crop (red boxes; on the right edge of the equirectangular image).

I also provide the user with tunable parameters to increase the flexibility of the tracker to footage of various resolution, quality, and camera movement:

- The user can determine the size of the window in which the tracker looks for a feature match in the next frame. This should be chosen according to the image size and level of detail. For a resolution of 1920×960 pixels, I set the square window size to 35 pixels.

- A minimum distance between features ensures an even spread across the image: if any two features are closer than this distance, then the shorter trajectory is terminated. I set this distance to 25 pixels.
- If the number of good feature trajectories drops below a threshold (270 in my case with 300 initial features), new features are created for tracking, so that they obey the user-defined minimum distance mentioned previously.
- Due to the small baseline between consecutive video frames, only every n^{th} frame (where n is the *keyframe offset*) is taken to estimate the initial camera poses. In most of the experiments, I set the offset to 5 frames. However, for some sequences with particularly slow camera movement, I observe an improvement to the accuracy of the final camera path when I set the keyframe offset to 10 frames.

For larger video resolutions (3840×1920 pixels), the window size and the minimum distance between the features are doubled to 70 and 50 pixels, respectively.

Once the 360° feature tracking is complete, I convert the 2D image coordinates of all features in the equirectangular image to their corresponding 3D unit direction vectors within the camera coordinate system. Let (u, v) be the normalised image coordinates of a feature corresponding to a point \mathbf{P} in 3D space, with the origin at the top left. The projection \mathbf{D} of the point \mathbf{P} on the unit sphere is then equivalent to the unit direction vector from the centre of the sphere, \mathbf{O} , towards \mathbf{P} :

$$\mathbf{D} = \frac{\mathbf{P} - \mathbf{O}}{\|\mathbf{P} - \mathbf{O}\|}. \quad (5.1)$$

I calculate the direction \mathbf{D} by mapping the image coordinates from the equirectangular image into the spherical domain. First, I convert the image coordinates (u, v) to spherical coordinates (ϕ, θ) (see Figure 2.23):

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{\pi}{2} - 2\pi u \\ \pi v \end{bmatrix}, \quad (5.2)$$

where azimuth $\phi \in [-\frac{3}{2}\pi, \frac{1}{2}\pi]$, elevation $\theta \in [0, \pi]$, and radius = 1. Then, I convert from spherical to Cartesian coordinates (in the local camera coordinate system: *x-left, y-back, z-up*) using

$$\mathbf{D} = \begin{bmatrix} \sin \theta \cos \phi \\ \sin \theta \sin \phi \\ \cos \theta \end{bmatrix}. \quad (5.3)$$

5.1.2 Epipolar Geometry

For two perspective cameras, exploiting epipolar geometry for SfM requires estimating an intrinsic matrix \mathbf{K} per camera which calibrates the field of view, and the fundamental matrix \mathbf{F} which describes the relations between the positions of points observed by both cameras. That leads to estimating the essential matrix \mathbf{E} which relates the positions and rotations of the cameras:

$$\mathbf{E} = \mathbf{K}_2^\top \mathbf{F} \mathbf{K}_1. \quad (5.4)$$

However, in the case of omnidirectional cameras, Chang and Hebert [13] noticed that both cameras can be treated as calibrated, because the positions of the observed points depend solely on relative rotation and translation between the two cameras (the field of view is fixed at 360°). Thereby, the essential matrix takes over the function of fundamental matrix, and so I only need to estimate the essential matrix to exploit epipolar geometry (see Figure 5.3).

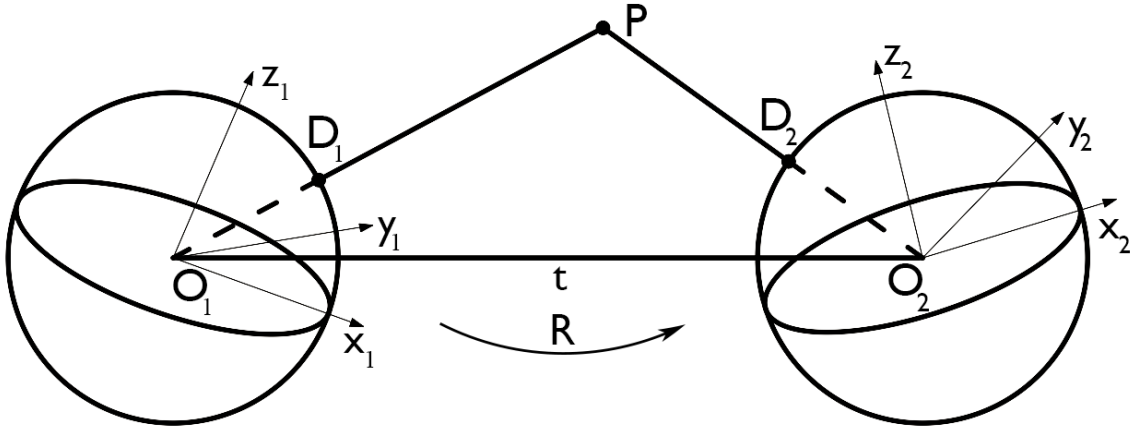


Figure 5.3: Epipolar geometry for omnidirectional cameras. World point P projects via camera centres O_1, O_2 onto sphere points D_1, D_2 , with each in its respective coordinate system $(x_1, y_1, z_1), (x_2, y_2, z_2)$. These systems are related by a translation vector t and a rotation matrix R .

If P is a point in 3D space and D_1, D_2 are its projections on two unit spheres representing the omnidirectional cameras, each with its associated local coordinate system, and a centre in, respectively, O_1 and O_2 , then all points P, D_1, D_2, O_1, O_2 are coplanar. Therefore,

$$\overline{O_2O_1} \times \overline{O_2D_1} \cdot \overline{O_2D_2} = 0, \quad (5.5)$$

which is equivalent to

$$\mathbf{O}_1^{(2)} \times \mathbf{D}_1^{(2)} \cdot \mathbf{D}_2 = 0, \quad (5.6)$$

where $\mathbf{O}_1^{(2)}$ denotes the centre of the first camera in the coordinate system of the second camera. Thus,

$$\mathbf{O}_1^{(2)} = \mathbf{R} \cdot \mathbf{O}_1 + \mathbf{t} = \mathbf{t} \quad (5.7)$$

$$\mathbf{D}_1^{(2)} = \mathbf{R} \cdot \mathbf{D}_1 + \mathbf{t}, \quad (5.8)$$

which leads to epipolar constraint:

$$\mathbf{D}_2^\top \mathbf{E} \mathbf{D}_1 = 0, \quad (5.9)$$

where \mathbf{E} is the essential matrix.

I estimate the essential matrix between every pair of consecutive frames using the eight-point algorithm adapted to 360° images [13]. First, I construct 3×3 matrices $\mathbf{M}^i = \mathbf{D}_1^i \mathbf{D}_2^{i\top}$ for each point correspondence i , where \mathbf{D}_1^i and \mathbf{D}_2^i are the projections of the 3D point \mathbf{P}^i on the two unit spheres representing the two cameras, as described in Section 5.1.1. Then, I stack the columns of the matrix \mathbf{M}^i into a column vector \mathbf{u}^i , and stack all of these vectors into an $n \times 9$ matrix $\mathbf{U} = [\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^n]^\top$, where n is the number of points. This matrix \mathbf{U} satisfies the linear system of epipolar constraints:

$$\mathbf{U} \mathbf{e} = 0, \quad (5.10)$$

where \mathbf{e} is the 9×1 vector constructed by stacking the columns of the essential matrix \mathbf{E} . I solve Equation 5.10 for \mathbf{e} using singular value decomposition (SVD).

Next, I decompose the essential matrix into a rotation matrix \mathbf{R} and a translation vector \mathbf{t} in the same way that it is usually decomposed for pinhole cameras [40]. This obtains the rigid-body transformation between the two cameras:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}. \quad (5.11)$$

In order to do that, I perform SVD, $\mathbf{E} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$, on the essential matrix (if $\det(\mathbf{U}) < 0$, $\mathbf{U} = -\mathbf{U}$ and if $\det(\mathbf{V}^\top) < 0$, $\mathbf{V}^\top = -\mathbf{V}^\top$) and set:

$$\mathbf{t}_1 = \mathbf{U} \mathbf{S} \mathbf{W} \mathbf{U}^\top \quad (5.12)$$

$$\mathbf{t}_2 = \mathbf{U} \mathbf{S} \mathbf{W}^{-1} \mathbf{U}^\top \quad (5.13)$$

$$\mathbf{R}_1 = \mathbf{U} \mathbf{W} \mathbf{V}^\top \quad (5.14)$$

$$\mathbf{R}_2 = \mathbf{U} \mathbf{W}^{-1} \mathbf{V}^\top, \quad (5.15)$$

where

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.16)$$

This produces four possible combinations of translations and rotations between the two cameras. I compare these solutions by triangulating all points, computing their reprojection errors, and selecting the decomposition with the smallest average reprojection error. Due to the small camera baselines between consecutive video frames, I use every fifth frame instead.

5.1.2.1 Triangulating 3D Points

Given the relative pose (\mathbf{R}, \mathbf{t}) between the two cameras, I can reconstruct the 3D position of the point \mathbf{P} observed at \mathbf{D}_1 in the first camera and at \mathbf{D}_2 in the second camera. For this, I use the midpoint triangulation method [39], as discussed by Ma et al. [64] in the context of 360° cameras. I wish to find the ‘intersection’ of the rays coming from the centres of the cameras, \mathbf{O}_1 and \mathbf{O}_2 , towards the observed directions to the point (Figure 5.3).

Without loss of generality, I assume that the first camera is centred at the origin and aligned to the global coordinate axes, so that the ray from the camera centre \mathbf{O}_1 and the direction \mathbf{D}_1 can be parametrised using $a \cdot \mathbf{D}_1$ for $a > 0$. The position of the second camera \mathbf{O}_2 is given by the translation vector \mathbf{t} in terms of the first camera’s coordinate system. After transforming the direction \mathbf{D}_2 from the coordinate system of the second camera to the first, I can write the ray corresponding to the second camera using $\mathbf{t} + b \cdot \mathbf{R} \mathbf{D}_2$ for $b > 0$. However, inaccuracies in point correspondences due to image noise and resolution limits may cause the two rays not to intersect, so I compute the midpoint of their minimum distance:

$$\operatorname{argmin}_{a,b} \|a \cdot \mathbf{D}_1 - b \cdot \mathbf{R} \mathbf{D}_2 - \mathbf{t}\|, \quad (5.17)$$

with the optimal solution given by:

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{t}, \text{ where } \mathbf{A} = [\mathbf{D}_1, -\mathbf{R} \mathbf{D}_2]. \quad (5.18)$$

The reconstructed point is located in the middle of this distance:

$$\mathbf{P} = \frac{\hat{a} \cdot \mathbf{D}_1 + \hat{b} \cdot \mathbf{R} \mathbf{D}_2 + \mathbf{t}}{2}. \quad (5.19)$$

5.1.2.2 Reprojection error

The 3D space is projected into the equirectangular image in a highly non-linear way, and pixels in different parts of the equirectangular image cover potentially different solid angles of the 3D space. Therefore, the distance in pixels in the equirectangular domain is not a meaningful measure of the reprojection error, and so I must compute error in the spherical domain.

Pagani and Stricker [76] described and compared three possible types of reprojection error in the spherical domain, based on Euclidean, geodesic, and tangential distances, respectively (Figure 5.4):

$$\varepsilon_e = \left\| \frac{\mathbf{P}}{\|\mathbf{P}\|} - \mathbf{D}_i \right\| \quad (5.20)$$

$$\varepsilon_g = \cos^{-1} \left(\frac{\mathbf{D}_i^\top \mathbf{P}}{\|\mathbf{P}\|} \right) \quad (5.21)$$

$$\varepsilon_t = 2 \tan \frac{\varepsilon_g}{2} = 2 \sqrt{\frac{1 - \frac{\mathbf{D}_i^\top \mathbf{P}}{\|\mathbf{P}\|}}{1 + \frac{\mathbf{D}_i^\top \mathbf{P}}{\|\mathbf{P}\|}}}. \quad (5.22)$$

Here, \mathbf{P} is the 3D point in the coordinate system of the unit sphere i (centred at the origin), and \mathbf{D}_i is the vector on the sphere representing the observation of \mathbf{P} in frame i as described in Section 5.1.1.

In their work, Pagani and Stricker tested these three errors in optimising the final camera pose, depending on the angle between the 3D point and its observation. The differences in performance become noticeable, especially for bigger angles. The error measures are ranked as follows, according to the decreasing residual error: the Euclidean distance, the geodesic distance, then the tangential distance. The test results in a recommendation to use the tangential distance as an error measure in optimisation in pose estimation for spherical cameras. I follow this recommendation, and I use the tangential reprojection error in the cost function in the bundle adjustment step of my algorithm (Section 5.1.4).

However, I do not use it to choose the correct camera extrinsics from the four solutions obtained from decomposing the essential matrix. In this case, I use the simpler Euclidean reprojection error. It is the fastest of the three to compute, and, even if the tangential distance scales better in general, it produces an infinite error for outliers reconstructed on the opposite side of the sphere. It makes it better for individual points (as in bundle adjustment), but it does not perform well for sets of points, where the mean reprojection error counts more than the individual ones.

For each of the four solutions obtained from decomposing the essential matrix (as per Equation 5.11), I triangulate all point correspondences, compute the mean Euclidean reprojection error ε_e across both cameras, and select the minimum as the best solution.

Finally, some outlier points may have been triangulated on the wrong side of the sphere. To cull these, I set a reprojection error threshold between $\varepsilon_e = 0$ (no error) and $\varepsilon_e = 2$ (the opposite side of the unit sphere). I experimentally discard $\varepsilon_e > 0.5$ for this task.

5.1.3 Multi-view Geometry

After calculating the relative camera pose (\mathbf{R}, \mathbf{t}) between each pair of consecutive frames, all poses need to be converted into a consistent global coordinate system. Without loss

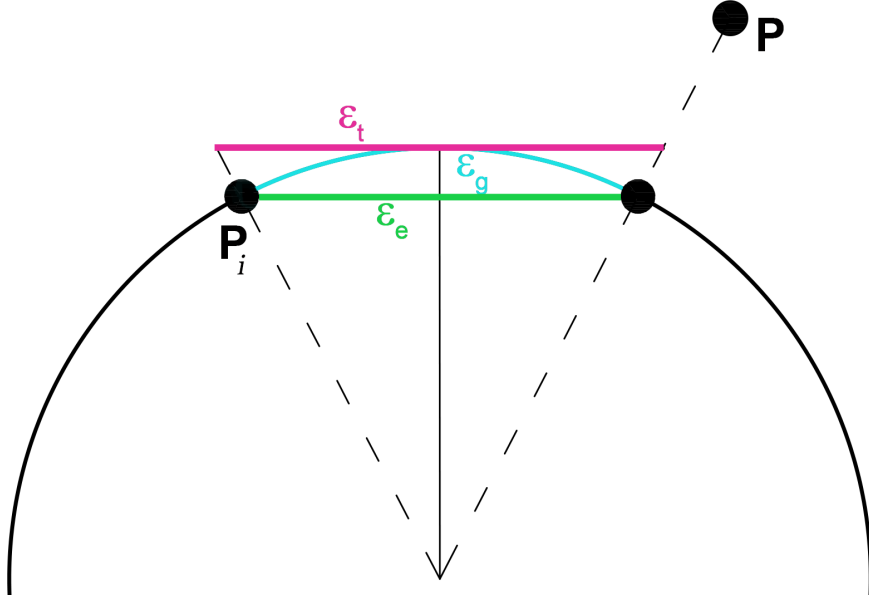


Figure 5.4: Different types of reprojection error on the unit sphere [76]: Euclidean distance ε_e , geodesic distance ε_g , and tangential distance ε_t .

of generality, I choose the coordinate system of the first video frame for this purpose, as any pose estimation is ultimately ambiguous up to a similarity transform [40]. I unify the coordinate systems sequentially by updating the poses of cameras in the following way:

$$\mathbf{R}_{\text{curr}} = \mathbf{R}_{\text{rel}} \mathbf{R}_{\text{prev}}, \quad (5.23)$$

$$\mathbf{t}_{\text{curr}} = \mathbf{R}_{\text{prev}} \mathbf{t}_{\text{rel}} + \mathbf{t}_{\text{prev}}, \quad (5.24)$$

where \mathbf{R}_{curr} , \mathbf{t}_{curr} , \mathbf{R}_{prev} , \mathbf{t}_{prev} are the current and previous-frame rotation and translation of the camera in global coordinate system, and \mathbf{R}_{rel} , \mathbf{t}_{rel} are relative rotation and translation between the current and the previous keyframe. At this point, the translation between two cameras is calculated up to scale, which means that the relative translation between each pair of consecutive frames is a unit vector. This is corrected in the next step of my approach: bundle adjustment.

5.1.4 Bundle Adjustment

I jointly optimize the camera motion and scene structure globally using bundle adjustment [106] for 360° cameras. However, due to the small camera baseline between consecutive video frames, direct global optimisation of all poses would be unstable as the initialisation is unreliable. Instead, I implement *hierarchical bundle adjustment*: (1) I perform bundle adjustment only for *keyframes* placed every five frames, and (2) I use the reconstructed keyframe poses and scene structure to initialize a second bundle adjustment pass with

all video frames. This produces the final camera motion path and scene structure reconstruction. Section 5.2.3 contains a detailed comparison of accuracy and execution times for one-step and hierarchical bundle adjustment to justify this design decision.

My implementation of 360° bundle adjustment uses the Ceres solver library [1], which supports arbitrary camera models and cost functions, and provides automatic differentiation. My camera model comprises seven extrinsic parameters (four for a rotation quaternion and three for a translation vector), and the cost function to be minimized is the average tangential reprojection error (Equation 5.22), which has shown the best convergence behaviour among the three reprojection errors covered in Section 5.1.2.2 [76]. I wrap the reprojection error within a robust Huber loss function $\rho(\cdot)$ to reduce the influence of outliers in my bundle adjustment objective:

$$\operatorname{argmin}_{\{\mathbf{P}_p\}, \{\mathbf{C}_c\}} \frac{1}{2} \sum_p \sum_c V_c^p \cdot \rho(\varepsilon_t^2(\mathbf{P}_p, \mathbf{C}_c, \mathbf{D}_c^p)), \quad (5.25)$$

where p iterates over all 3D points and c over all cameras, $V_c^p \in \{0, 1\}$ represents if point p is visible in camera c , ε_t is the tangential reprojection error (Equation 5.22), \mathbf{P}_p is the 3D position of point p , \mathbf{D}_c^p is its projection in camera c , $\mathbf{C}_c = [q_w, q_x, q_y, q_z, t_x, t_y, t_z]$ parametrises the pose of camera c , and ρ is the Huber loss of the squared residual [1] with scaling factor $\delta = 0.007$:

$$\rho(s) = \begin{cases} s & s \leq \delta \\ \delta(2\sqrt{s} - \delta) & s > \delta \end{cases}. \quad (5.26)$$

For the first bundle adjustment pass, I use the keyframe poses computed in Section 5.1.3 as initial poses. As initial structure, I use the points triangulated from the first pair of keyframes in which each trajectory is observed. For the second bundle adjustment pass, I interpolate the camera poses for in-between frames from the keyframe poses (using linear interpolation for translation vectors and spherical linear interpolation (Slerp) for quaternion interpolation [17]), and use the previously reconstructed structure for initialization.

5.2 EXPERIMENTS

After both passes, I have recovered a pose for every camera frame and a set of 3D world points (see Figure 5.7). This allows me to match virtual camera views to real camera views.

5.2.1 Synthetic Data

A Toy Example. Firstly, I tested the algorithm on a toy example created in MATLAB. The example contains sets of points organised into simple parallel and perpendicular planes and two virtual spherical cameras represented by their centres, which correspond to two

video frames. Each point is projected on both cameras' spherical surfaces, which provides direction vectors \mathbf{D}_1 and \mathbf{D}_2 for epipolar geometry (Figure 5.3). Figure 5.5 shows the visualised results of reconstruction compared with the ground truth.

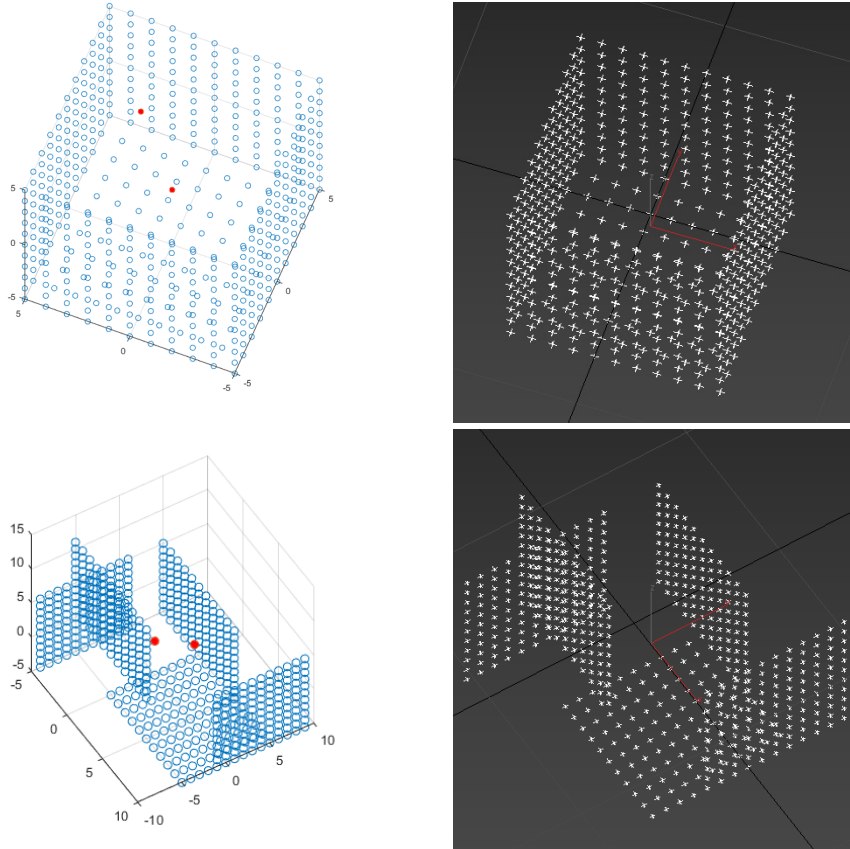


Figure 5.5: **Left:** Synthetic data generated in MATLAB (ground truth). Red dots represent the cameras' centers and blue circles represent 3D points. **Right:** Points reconstructed by the omnidirectional structure-from-motion algorithm described in this chapter and visualised in 3D software. Structure from motion can reconstruct the geometry only up to scale.

A CG Scene. The next synthetic example to test my algorithm is a CG scene created in Unity and rendered as a 360° video in 60 fps. The scene is set inside a supermarket with a camera moving slowly between the shelves. The video rendered from a virtual camera rig is free from stitching errors usually present in the video from a real camera and contains no camera rotation. The camera movement is fully controlled and may thus serve as a ground truth.

Figure 5.6 depicts the reconstruction result with the point cloud and camera path viewed from different angles and the sample input frame.

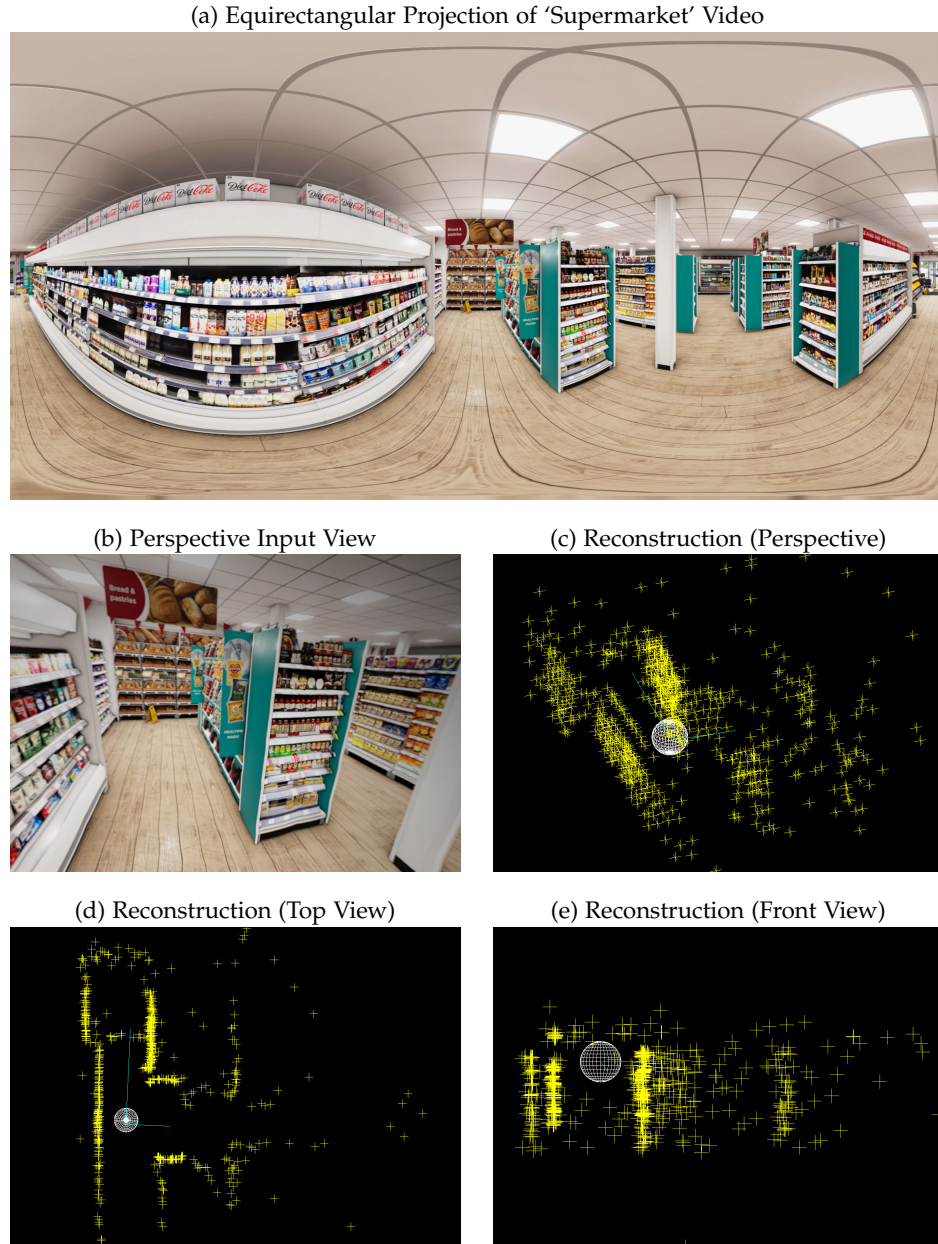


Figure 5.6: Reconstructed scene. (a) Input frame from synthetic omnidirectional video. (b) A perspective view of the input omnidirectional video. (c, d, e) Perspective, top and front views of the reconstructed camera path (cyan), with the view in (b) shown as white wire-frame sphere, and sparse scene geometry (yellow points).

Sequences from Facebook Replica and TUM RGBD SLAM Datasets. I test the pipeline on seven synthetic video sequences using a 360° renderer for the Facebook Replica dataset [95]. The first four sequences have camera paths which are synthetic: a straight line and a circle, with and without structured random jitter to simulate a handheld camera. The last three sequences have camera paths from the TUM RGBD SLAM dataset [96], which I assume to be representative of real-world camera motion. Table 5.1 shows the average Euclidean error between my reconstruction and ground-truth camera paths. Before calculating the error, the reconstructed trajectory undergoes the Procrustes superimposition to align it with the ground truth and scale it accordingly. The structure-from-motion pipeline performed well on video sequences with no or slight vertical camera rotation, but the last sequence with the camera rotating through the poles revealed its limitations and produced high average error. With such types of rotation, features are heavily distorted and are frequently lost by the tracker. This, in turn, leads to only a small number of good features for reconstruction and so inaccurate estimation of camera extrinsics between the two keyframes.

Table 5.1: Quantitative evaluation on synthetic 360° videos rendered from the Facebook Replica dataset [95], using the average Euclidean error (in millimeters) between reconstruction and ground-truth camera path. The first four video sequences use synthetic camera paths, with the other three camera paths coming from the TUM RGBD SLAM dataset [96]. The high reconstruction error in the last sequence is caused by camera rotation over the poles, which leads to short feature trajectories as the tracker drops heavily distorted points. All sequences but last two have the keyframe offset set to 5 frames, and the last two sequences have it set to 10.

Video	#frames	Error (mm)
Straight line	446	5.4 ± 2.0
+ jitter	446	3.9 ± 1.6
Circular pan	716	11.0 ± 5.9
+ jitter	716	22.8 ± 16.1
TUM path 1	696	2.8 ± 0.9
TUM path 2	696	7.2 ± 0.5
TUM path 3	996	225.6 ± 171.1

5.2.2 Real Data

I tested my approach with handheld content recorded from ‘Ricoh R’ and ‘Insta360 ONE X’ 360° video cameras. Table 5.2 contains the parameters of the tested sequences.

Figure 5.7 shows example input video frames in equirectangular projection, as well as several views of the reconstructed camera motion and scene structure. The reconstructions

match the input environments well, as can be seen when comparing them to a perspective sub-view of the omnidirectional input image.

Table 5.2: Real 360° video sequences used to test the omnidirectional structure from motion approach.

Sequence	Resolution	fps	Camera model
‘Crescent’	1920×960px (pixel aspect ratio corrected from 1920×1080px)	29.97	Ricoh R
‘Parade’	3840×1920	29.97	Insta360 ONE X
‘Sicilian Avenue’	3008×1504	100	Insta360 ONE X
‘The Circus’	3840×1920	50	Insta360 ONE X

5.2.3 Comparison Between Hierarchical and One-Pass Bundle Adjustment

Table 5.3 shows compared accuracy (if there was a ground truth available) and execution times for one-pass and hierarchical bundle adjustment. They were performed on synthetic and real video sequences described in previous sections. One-pass bundle adjustment was

Table 5.3: Comparison of execution times and accuracy between one-pass and hierarchical bundle adjustment.

Video	#frames	Time overall one-pass BA (s)	Time overall hierarchical BA (s)	Error one-pass BA (mm)	Error hierarchical BA (mm)
Straight line	446	679.0	447.2	5.4 ± 1.9	5.4 ± 2.0
+ jitter	446	728.0	511.4	4.2 ± 2.0	3.9 ± 1.6
Circular pan	716	1167.5	634.0	14.2 ± 16.9	11.0 ± 5.9
+ jitter	716	574.2	527.2	24.1 ± 18.5	22.8 ± 16.1
TUM path 1	696	1254.8	916.1	4.7 ± 1.8	2.8 ± 0.9
TUM path 2	696	855.8	939.3	8.8 ± 8.0	7.2 ± 5.0
TUM path 3	996	1116.9	772.9	498.2 ± 252.3	225.6 ± 171.1
Parade	921	977.1	627.9	n/a	n/a
Sicilian Avenue	806	1252.0	966.3	n/a	n/a
The Circus	1001	1041.3	799.7	n/a	n/a
Crescent	751	724.3	668.2	n/a	n/a

faster in just one case, and hierarchical bundle adjustment proved to be more accurate in all sequences with available ground truth. These results confirm that the two-pass hierarchical bundle adjustment outperforms the single-pass one in both accuracy and speed, and was a good design choice.

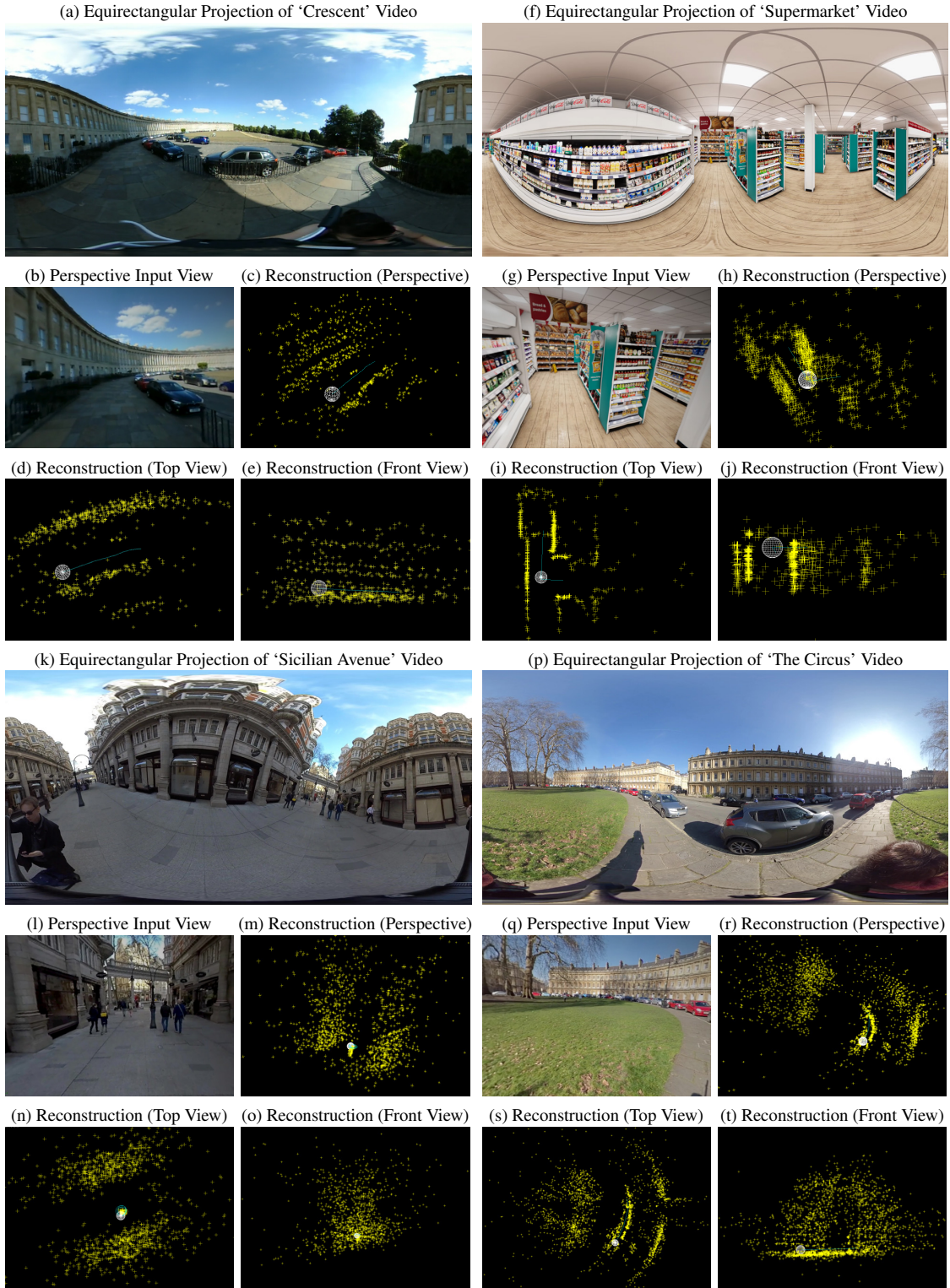


Figure 5.7: Results of my omnidirectional structure-from-motion pipeline for real videos. (a, f, k, p) Input frames from omnidirectional videos. (b, g, l, q) A perspective view of the input frames. (c-e, h-j, m-o, r-t) Perspective, top and front views of the reconstructed camera path (cyan), with the view in (b, g, l, q) shown as a white wire-frame sphere, and sparse scene geometry as yellow points.

5.3 DISCUSSION

My approach comprises a structure-from-motion pipeline for omnidirectional videos that works directly on stitched equirectangular video and reconstructs both camera motion and the sparse structure of the environment. The reconstructions match the input environments well, as can be seen when comparing them to a perspective sub-view of the omnidirectional input image.

However, there are also limitations to be considered. In feature tracking, my template-based tracker becomes lost when the feature distortion becomes too large towards the poles in the equirectangular image. This results in shorter trajectories and incorrectly reconstructed points caused by the tracker picking the same or spatially-very-similar features to those lost in the previous frame.

REAL-TIME VIRTUAL OBJECT INSERTION FOR MOVING 360° VIDEOS

In Chapter 5, I addressed the first of the three challenges associated with inserting virtual objects into real footage, camera tracking, in the context of 360° videos. In this chapter, I would like to address the remaining two, the illumination estimation and the rendering and compositing of the inserted objects, which complete the virtual object insertion pipeline for moving 360° videos.

This work extends the dynamic mixed-reality compositing pipeline described in Chapter 3 to 360° videos. In the pipeline, omnidirectional structure from motion replaces off-the-shelf tracking software previously used for perspective cameras. There is also no need to capture additional environment maps for image-based lighting, due to the nature of 360° videos, where each frame provides an environment map.

Michiels et al. [69] designed a pipeline to insert virtual objects into 360° videos recorded with a moving camera, however, their insertions lack visual fidelity. This is due to utilising LDR environment maps for image-based lighting and the lack of shadows cast by virtual objects. The current state-of-the-art method, MR360 [80], provides a complete pipeline for inserting visually plausible CG objects into 360° videos with real-time user interaction, but only for static cameras (see Section 2.6.2).

My work contributes with combining the benefits of these two approaches. I present a pipeline that allows the virtual objects to be inserted into a video from a moving camera and interacted with in real time. For maximum visual fidelity, they cast shadows to their surroundings and are lit with 360° video frames converted to HDR images and used as environment maps (see Section 2.6.1). Instead of using a simple formula for inverse tone mapping for LDR to HDR conversion [47, 80], I decided to use one of the available state-of-the-art deep learning methods [27]. I also extended my pipeline with spatially-varying reflections for even more realistic insertions, which cannot be found in the other pipelines.

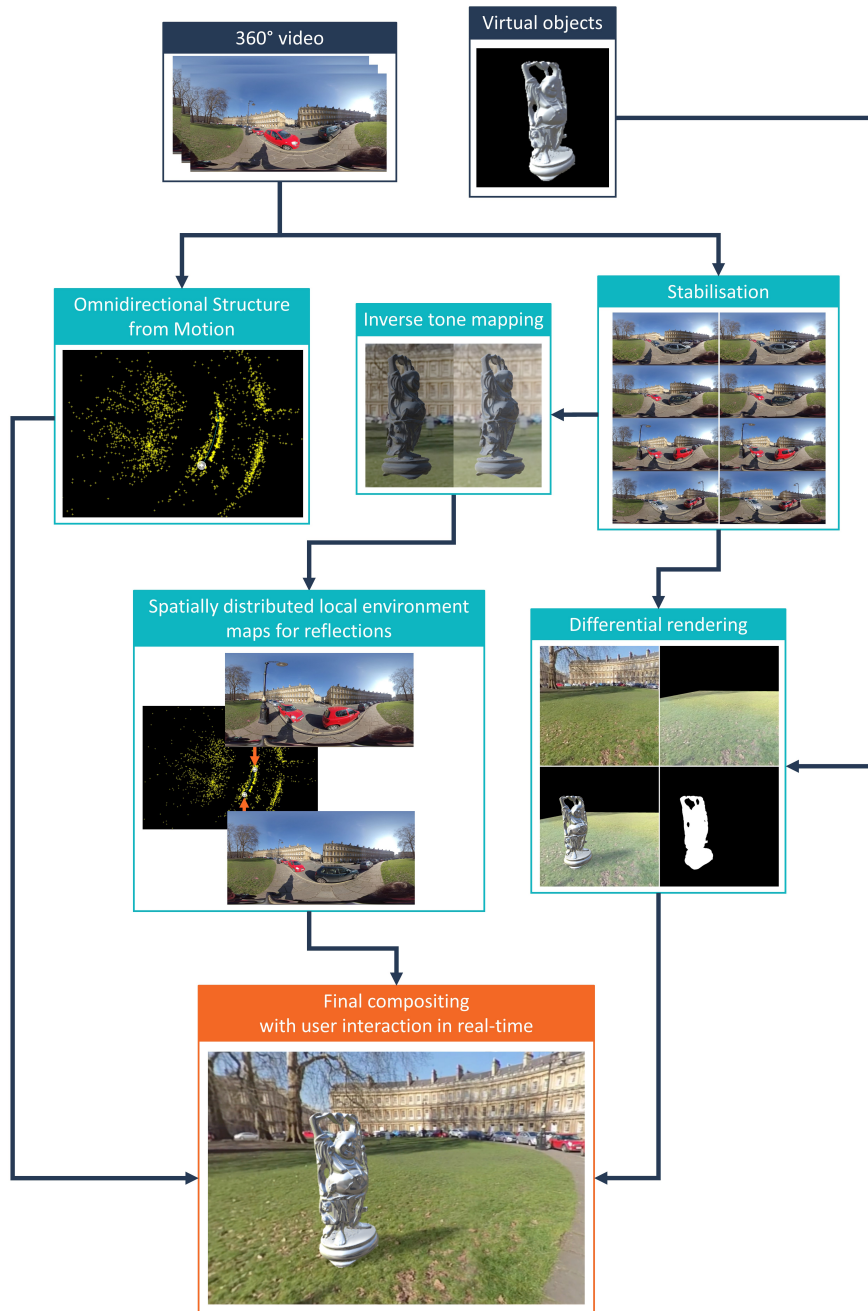


Figure 6.1: Overview of my approach for inserting virtual objects interactively into 360° videos, showing inputs (dark blue), processing steps (turquoise) and the result (orange). I employ omnidirectional structure from motion for camera tracking and 360° video stabilisation, improve image-based lighting using inverse tone mapping and spatially-distributed local environment maps, and use differential rendering for image-based shadowing. I implement my approach in the Unity game engine for final compositing and user interaction in real time.

6.1 VIRTUAL OBJECT INSERTION PIPELINE

High-quality virtual object rendering requires both illumination estimation to light the object plausibly, and real-time rendering to composite the objects with the video footage. I perform lighting estimation in a preprocess, and then use real-time rendering in the Unity engine for interactive applications.

For image-based lighting, first I stabilise the 360° video to align all environment maps with the Unity global coordinate system (Section 6.1.1). Then, I apply inverse tone mapping to the stabilised 360° video to recover HDR environment maps (Section 6.1.2). I import these into Unity together with the reconstructed camera path and 3D points from Section 5.1.4, so that they act as a local environment map for reflections (Section 6.1.4). In Unity, synchronisation between the camera animation and the camera footage is handled in an analogous way to the one described in Section 3.1.4 for perspective cameras. Finally, I implement differential rendering to enable casting of virtual shadows on top of the video footage (Section 6.1.5).



Figure 6.2: Omnidirectional video stabilisation. **Left:** input frames from a handheld camera moving along the circular path. **Right:** stabilised video frames with consistent horizon. In every frame, the forward and up vectors of the camera points in the same directions, respectively.

6.1.1 360° Video Stabilisation

The Unity rendering engine expects environment maps to be aligned to the global coordinate system. However, in general, this is not the case for an arbitrary 360° input video. Therefore, we need to stabilise the video to correctly orient the lighting and reflections for image-based lighting (Section 6.1.4). For each input video frame, I rotate the recovered camera in its opposite orientation, i.e., by \mathbf{R}^\top , so that all frames are aligned with the camera coordinate system of the first frame (which I consider to be the global coordinate system). This camera rotation corresponds to a resampling of the equirectangular image according to the rotation applied, for which I use bilinear interpolation. This eliminates shaky rotations and produces a view with a consistent horizon and up direction (Figure 6.2), and makes my video suitable for image-based lighting in Unity.

6.1.2 Inverse Tone Mapping

Image-based lighting with high-dynamic-range (HDR) environment maps visibly improves rendering results by overcoming muted low-contrast reflections [19]. Thus, I integrate inverse tone mapping into my approach, which aims to recover HDR images from low-dynamic-range (LDR) input images [25, 27, 67]. I use the approach of Endo et al. [27], which learns to estimate an exposure-bracketed set of images from a single input image via a deep convolutional neural network, and then merges this set of exposures into an HDR radiance map using Debevec and Malik’s approach [20]. Figure 6.3 shows a comparison of image-based lighting for low- and high-dynamic-range environment maps.



Figure 6.3: Comparison of image-based lighting with LDR (left) and HDR (right) environment maps based on the same input video frame. The HDR version is better exposed in shadows and highlights.

6.1.3 *Displaying 360° Videos in Unity*

In the spherical camera model, a sphere surrounding the camera centre replaces the image plane of a perspective camera, as described in Section 2.4.2. That induces 360° videos to be displayed on the spherical surface instead of on a rectangular plane.

Unity 2017 introduced a greater support for 360° video, including its display through a panoramic shader on the Skybox. It allows 360° videos to be displayed directly on the Skybox via a VideoPlayer component, which enables rendering a texture applied to the panoramic shader. While it simplifies the process and unifies the way standard and 360° videos are handled, the Skybox is not a desirable representation of a moving 360° camera. It is represented by a cubemap mapping of a sphere of, theoretically, infinite radius and therefore cannot model the distance between the camera and the virtual objects in the scene.

In this case, a better representation of a moving 360° camera is a simple sphere, whose pose can be animated to match the movement of the real camera. Video is displayed on the inner side of the sphere, using spherical texture mapping with flipped surface normals.

However, there seems to be an issue with Unity's spherical mapping caused by a low resolution default sphere mesh that is unable to generate a UV map of sufficient density¹. The particularly problematic areas are points directly above and below the centre, i.e., zenith and nadir, with mapping artifacts and poorly preserved straight lines. Recommended solutions include writing an equirectangular shader with inverted surface normals and applying it to the material, or creating a sphere with denser geometry, using an external 3D software, and importing it into Unity with baked UV mapping. Figure 6.4 compares the outputs of Unity's default spherical mapping, equirectangular shader and UV mapping imported with a dense sphere geometry from 3ds Max for two example scenes, the synthetic 'Supermarket' and the real 'The Circus'. Both solutions produce satisfactory results, and the small mapping artifacts still present in the virtual scene are most likely a result of the way the scene was rendered, and not the mapping itself.

¹ <https://forum.unity.com/threads/what-is-wrong-with-unitys-spherical-mapping-how-to-fix-it.321205/>, last accessed 31/07/2019.



(a) 'Supermarket' scene, nadir view



(b) 'Supermarket' scene, zenith view



(c) 'The Circus' scene, nadir view



(d) 'The Circus' scene, zenith view

Figure 6.4: Spherical texture mappings available in Unity, demonstrated on the example scenes: (a, b) 'Supermarket', nadir and zenith view, (c, d) 'The Circus', nadir and zenith view. **Left:** default spherical Unity mapping. **Middle:** equirectangular shader with flipped normals. **Right:** 3ds Max spherical mapping imported into Unity with a sphere object.

6.1.4 Image-based Lighting in Unity

After all preprocessing steps, I import the estimated camera path, scene points, and HDR video frames as environment maps into the Unity engine for real-time dynamic image-based lighting [19]. Unity 2018 offers two mechanisms to use environment maps: (1) as a *skybox* which represents distant illumination, equivalent to *distant scene* in Debevec’s light-based scene model [18] (see Section 2.3.0.1), and (2) as *reflection probes* which represent nearby illumination. Every rendered frame usually uses a single skybox, based on the current video frame during playback, but multiple reflection probes can be distributed throughout the environment at the same time to model spatially-varying illumination (Figure 6.7).

To improve rendering efficiency, I do not use all video frames as reflection probes, as this could easily comprise hundreds of HDR images with many corresponding to very similar locations in space and thus very similar lighting environments. Instead, I divide the camera path into a number of segments, as specified by the user, and place a reflection probe in each segment. At run time, Unity uses the skybox for distant image-based lighting and the nearest reflection probe to compute reflections (Figure 6.5).

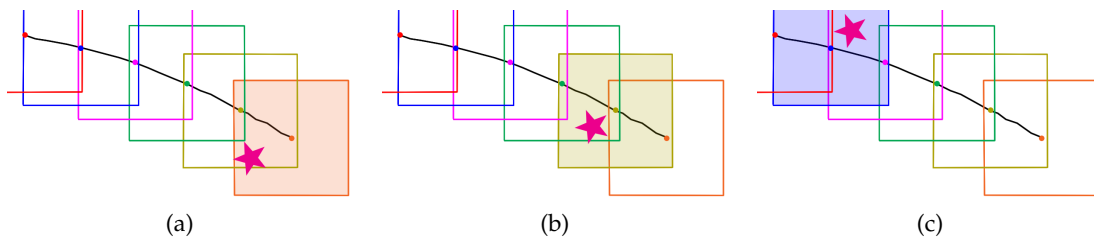


Figure 6.5: A symbolic top view of a scene. The black line represents the camera path. Each small circle is a reflection probe sampled from the frames along the path. A square surrounding each probe is its region of influence. If we place an object (here depicted by a star), its reflection map depends on its position within the scene. Also, the object takes reflections only from the closest probe, even if it is within the range of several of them.

6.1.5 Differential Rendering for Virtual Shadows

I described the differential rendering principles in Section 2.3.0.2 and the example of its use and implementation in Unity for perspective videos in Section 3.1.7. Its implementation for 360° videos is very similar as it does not depend on the input frame format. It rather is a rendering effect present in the view from a virtual perspective camera looking at the 360°

frame from the inside. Therefore, the final composite is described by the same Equation 2.7 from Section 2.3.0.2:

$$C = \alpha \cdot O + (1 - \alpha) \cdot (B + O - L). \quad (6.1)$$

The only difference is that the background image B is not the input video frame but the part seen by a virtual perspective preview camera. Local scene geometry such as the

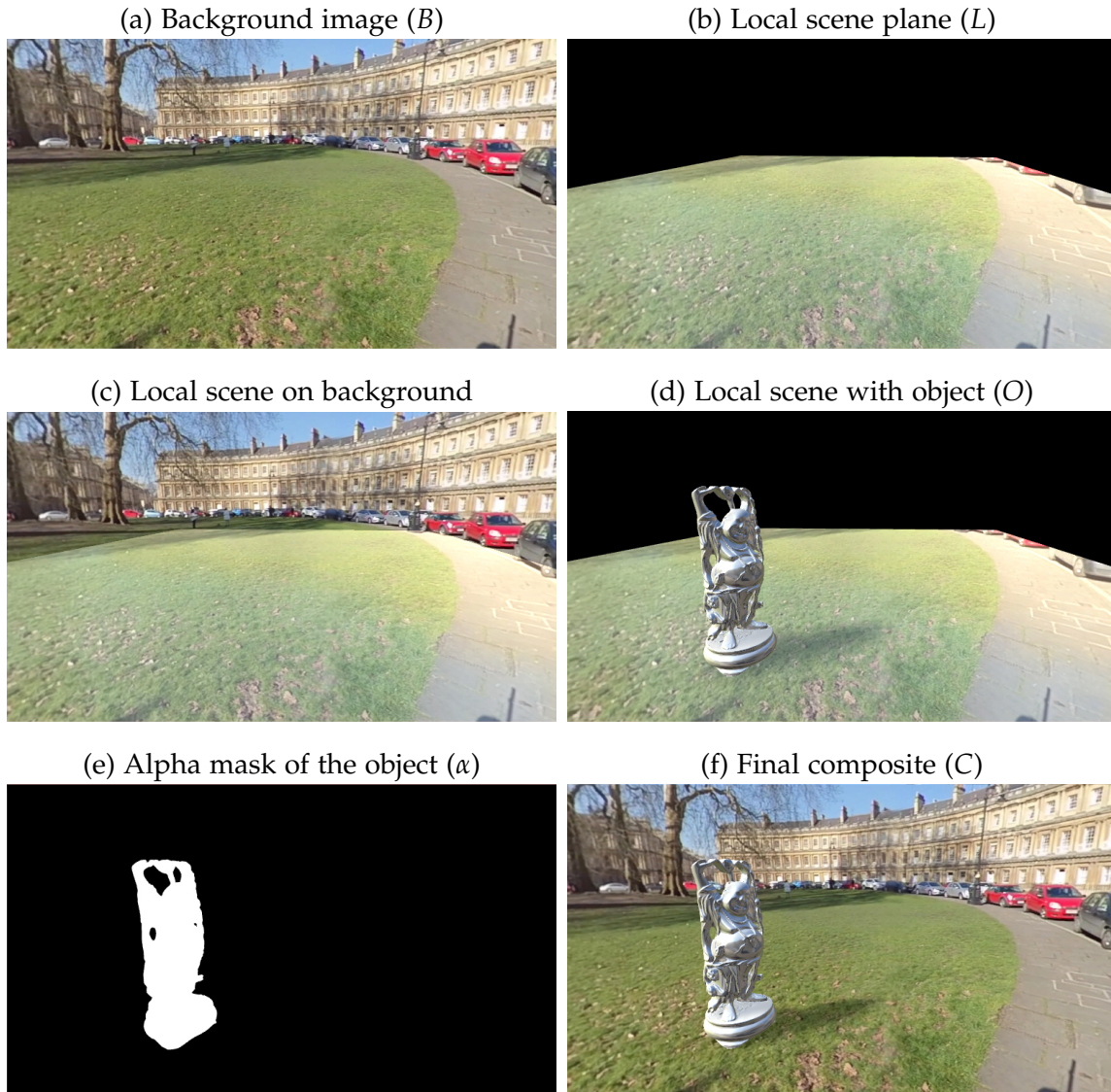


Figure 6.6: Components of differential rendering: **(a)** the background image B , **(b)** local scene reconstruction L using a plane, **(c)** L rendered on top of the background B , **(d)** the local scene with objects O , **(e)** the object's alpha mask α , and **(f)** the final composite C .

ground plane is modelled by the same script, which fits a plane automatically to selected points. The main light source, again, is placed manually.

Figure 6.6 illustrates the components of differential rendering for a scene with 360° video as a background.

6.2 EXPERIMENTS

First, I tested only spatially distributed local environment maps for reflections. I placed virtual mirror spheres in the scene at different distances from the camera and compared the reflections produced by two types of environment maps (Figure 6.7).

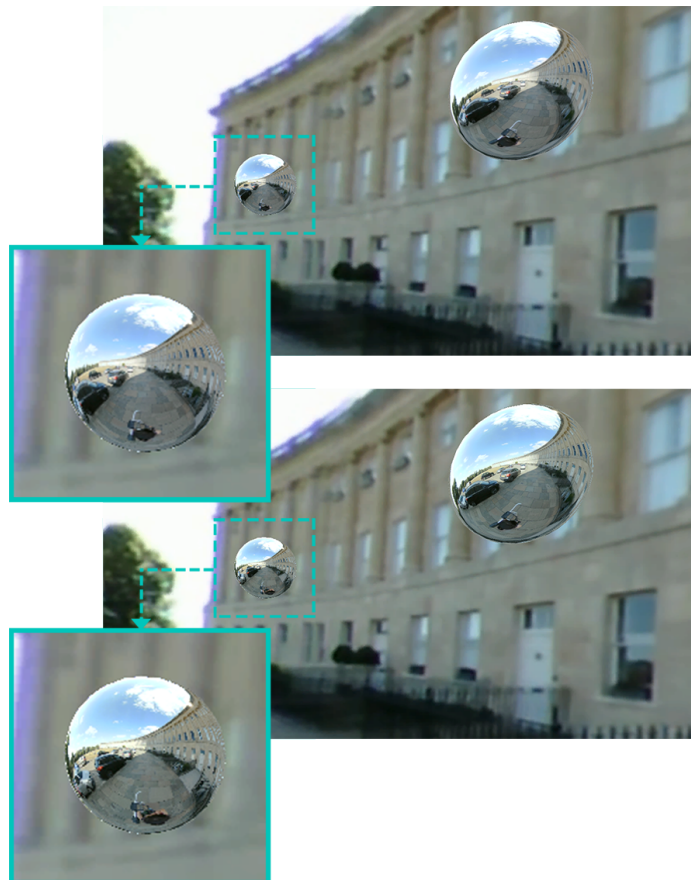


Figure 6.7: Virtual mirror spheres inserted at two different locations in the scene. **Top:** a single global environment map results in identical reflections in both mirror spheres, which is incorrect (e.g., the position of the black car on the left-hand side of the sphere between the near and far spheres). **Bottom:** Frames along the camera path used as environment maps, which produces visibly different reflections (e.g., see the black car on the left and the house entrance on the right).

Then, I inserted in the same scene more complex reflective objects (a dragon, a bunny, and a happy Buddha statue) and I enabled camera movement (Figure 6.8). I observed how reflections changed between different objects based on their placement in the scene and the camera position.

This test confirmed that with a single global environment map centred in the global coordinate system, reflections do not show the correct perspective. My set of spatially-varying reflection maps make virtual object reflections more convincing. Using this approach, results are most accurate when objects are placed *on* the original camera path; becoming less accurate as their distance from the camera path increases.

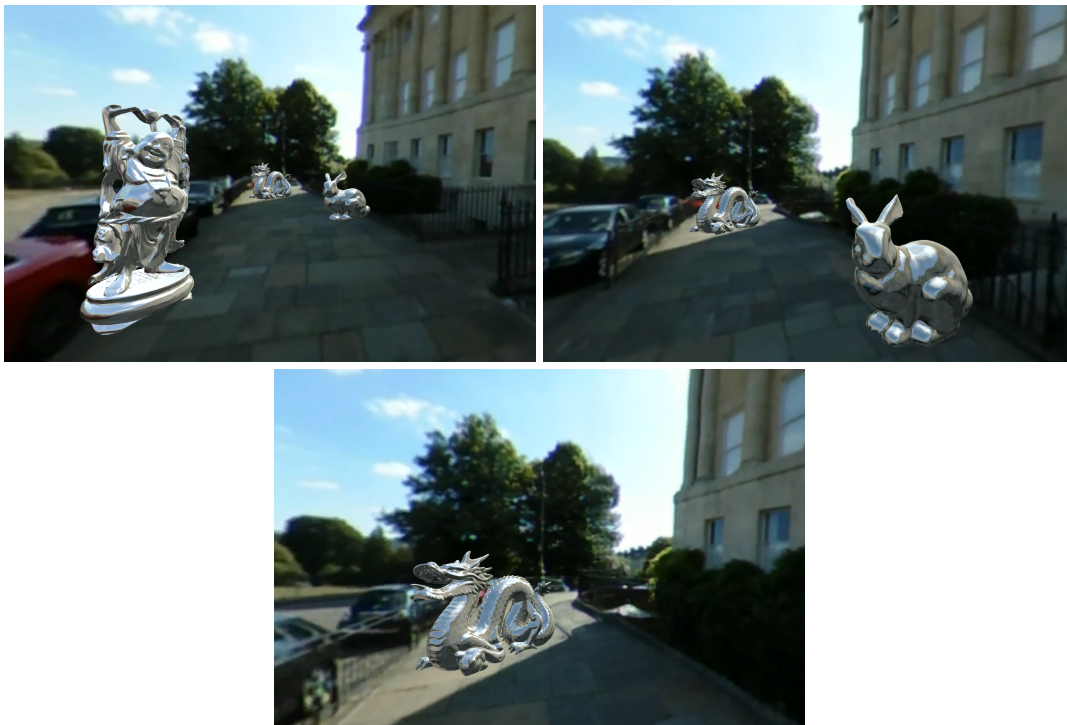


Figure 6.8: Computer-generated elements inserted into real video footage in real time with the camera moving forward. Notice how the red car is only reflected in Buddha, and not the bunny or dragon.

Next, I applied the whole virtual object insertion pipeline (Figure 6.1) to the 360° video sequences described in Chapter 5: ‘The Circus’, ‘Crescent’ and ‘Parade’ (real), and ‘Supermarket’ (synthetic). Figure 6.9 shows four examples of computer-generated objects inserted into these videos in real time. The inserted virtual objects can also be manipulated interactively, as I demonstrate in Figure 6.10. For example, objects can be moved, rotated and potentially animated, with real-time image-based lighting and shadowing.

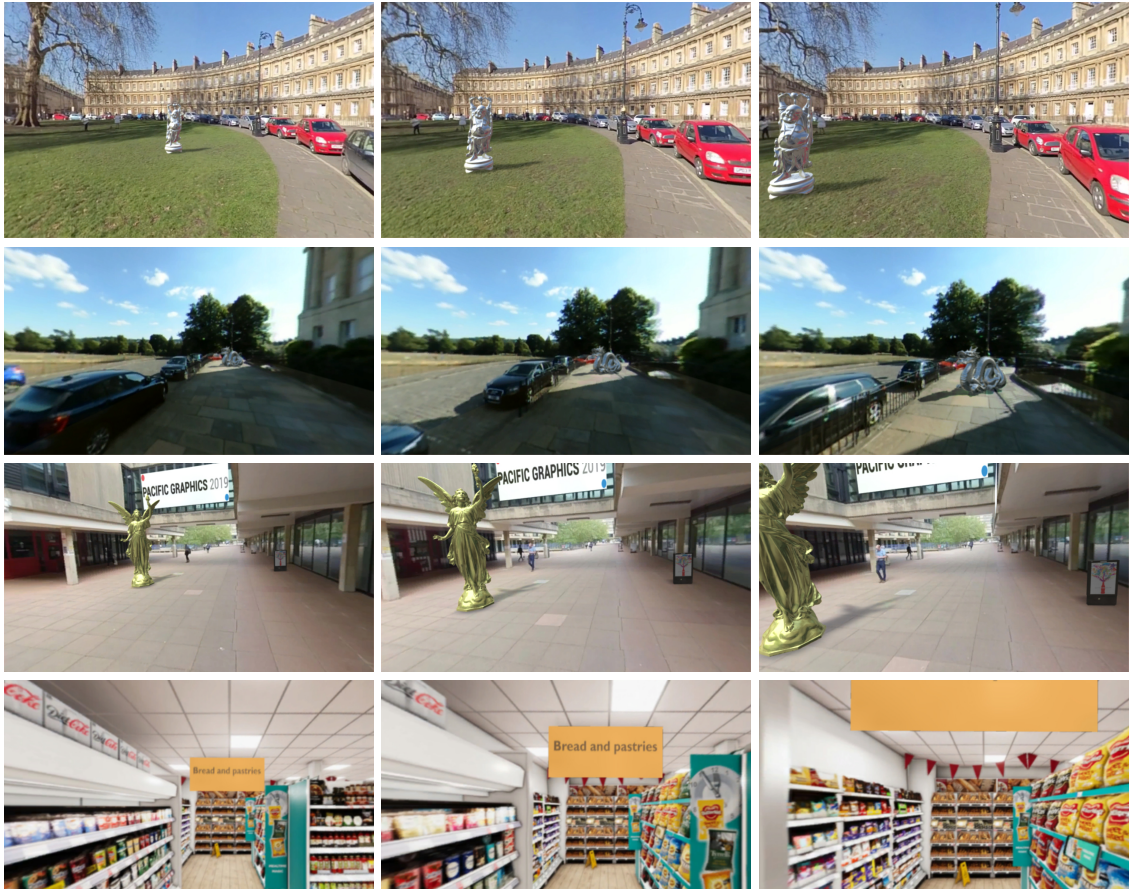


Figure 6.9: My approach enables real-time insertion of computer-generated objects into 360° video footage. **Left to right:** The camera is moving forward towards the inserted object. **Top:** A silver Buddha is inserted into ‘The Circus’ video (real scene). Note the reflection of the red car on the belly. **Middle top:** A silver dragon inserted into the ‘Crescent’ video (real scene). **Middle bottom:** A golden angel, a small poster stand and a conference banner are inserted into the ‘Parade’ video (real scene). **Bottom:** The billboard “Bread and pastries” is inserted into the ‘Supermarket’ video (synthetic scene).

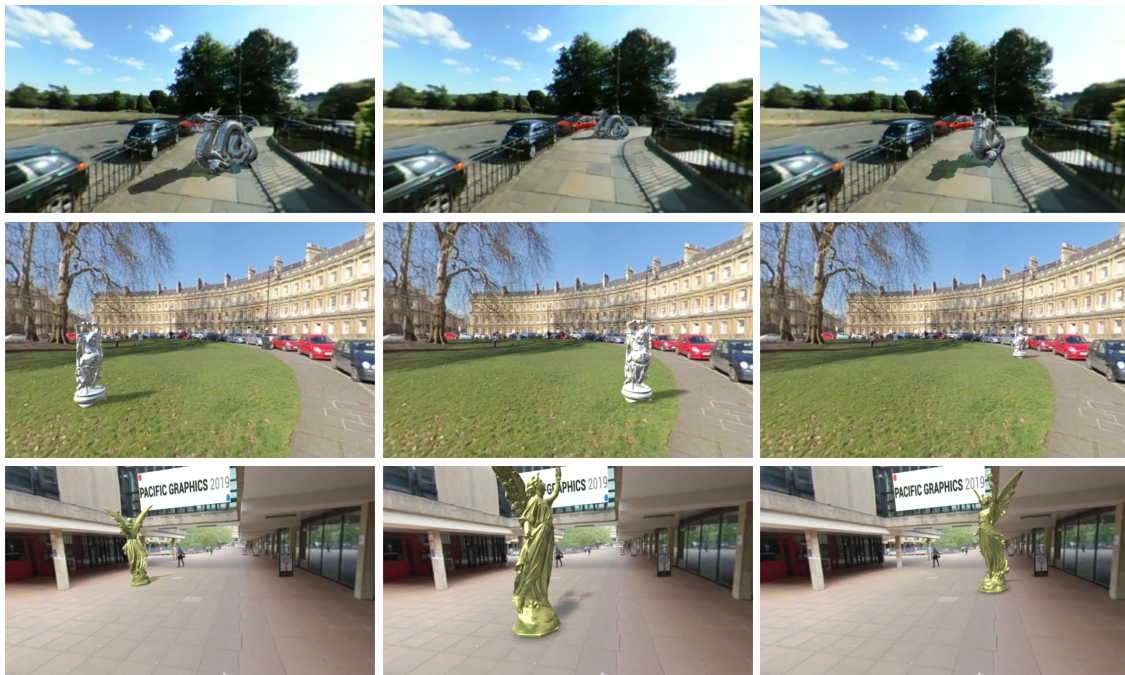


Figure 6.10: Computer-generated objects: the dragon (**top**), Buddha (**middle**) and the angel (**bottom**) can be manipulated interactively, including moving and rotating the CG objects.

Although it can also be observed in Figure 6.10, for better clarification, Figure 6.11 provides a closer look into how reflections of the inserted object change when it moves within the scene in real time.

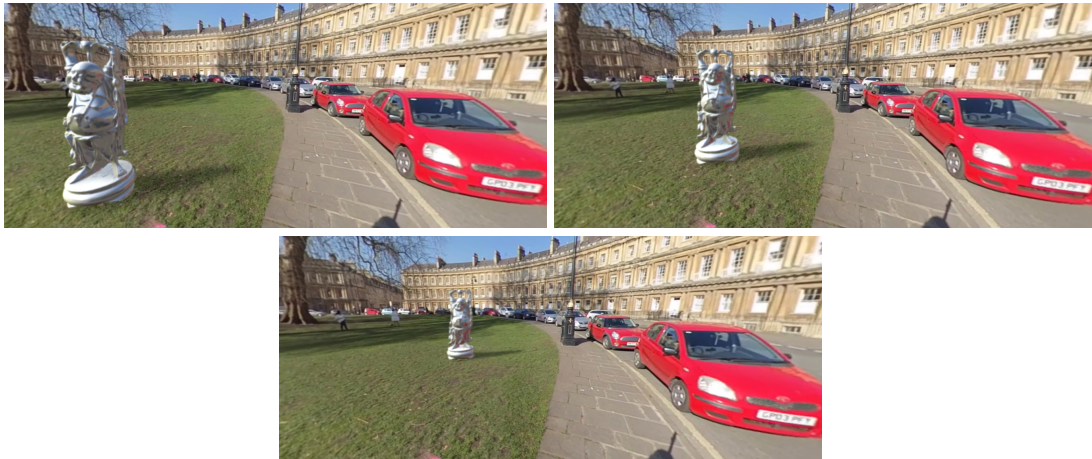


Figure 6.11: The Buddha statue moves around in ‘The Circus’ scene in real time, while the camera stays fixed in place. Note the change of reflections when it moves closer to red cars.

Figure 6.12 highlights another important part of my pipeline, which is using inverse tone-mapped environment maps in virtual object lighting. There is a noticeable difference between the look of inserted objects lit by LDR maps and those lit by HDR maps, in favour of the latter.



Figure 6.12: Comparison of lighting with LDR environment maps (**left**) and inverse tone-mapped HDR maps (**right**). HDR maps produce visually more accurate and vivid results, while lighting with LDR maps results in underexposed objects.

6.3 DISCUSSION

My approach brings higher quality real-time rendering to 360° virtual object insertion applications; however, challenges remain. In rendering, common problems of sampling in image-based rendering occur. I chose to pick the nearest probe for lighting, as interpolating between lighting environments results in ghosting artefacts. As probes are placed more

densely, this error decreases while memory and storage load increase. Distant probes are generally fine as light sources for low-frequency lighting but are more problematic for high-frequency reflection content of shiny materials. My approach also cannot support inter-reflections between the video and inserted virtual elements.

When retrieving HDR versions of LDR frames, Endo et al.'s inverse tone mapping [27] takes approximately 6 minutes per video frame of resolution 2048×1024 pixels (for an Intel Core i7 2.7 GHz processor with 16 GB RAM). This is very time-consuming for long video sequences, even if we assume that only a subset of the input frames is converted. In addition, if the frame size, both width and height, is not a power of 2, they have to be resized before and after conversion, which adds to the overall time.

Accurate shadowing of virtual objects requires a dense reconstruction of the world with a surface representation such as a mesh. My current approach only reconstructs sparse world points, and uses a single estimated point light source for shadowing. Recovering dense world geometry is a significant open problem, but would allow realistic world occlusion and shadowing of virtual objects.

CONCLUSIONS

In my work, I investigated methods for inserting computer-generated elements in a photorealistic way into real videos, which may be useful in creating video assets for market research surveys. Because such surveys usually require multiple versions of the same video, which differ just in details of inserted objects, the insertions should be real-time, according to the version that is currently required for display. The project was a collaboration with two market research companies, Checkmate VR and Dc-activ. With my industrial collaborators in mind, I focused on practical aspects of the task and on creating a complete compositing pipeline that could be used and tested by them in combination with their research survey building environment.

I began my research with standard perspective videos (Chapter 3). That allowed me to focus only on the real-time compositing in the Unity game engine, as camera tracking for standard videos has well-established solutions with plenty of off-the-shelf software available. Then, I extended the resulting dynamic compositing pipeline to spherical 360° videos, with image-based lighting and shadowing (Chapter 6), and with an addition of my design and implementation of omnidirectional camera tracking (Chapter 5).

In the remainder of this chapter, I review my original research objectives (listed in Section 1.2) and present the resulting contributions. I also suggest directions for future work. The chapter ends with my reflections on conducting research in a particular work environment that links academia to industry.

7.1 INSERTING CG OBJECTS INTO REAL FOOTAGE

My first and main research objective was as follows:

O1: To design a method to insert CG objects into real footage that works for both standard and spherical videos. The design process requires addressing the technical challenges related to camera tracking, lighting estimation and compositing in real time. To fulfil the plausibility requirement, all the insertions must be the photorealistic ones.

Being embedded in industry both set limitations as to the choice of tools to use and also focused my research on specific applications. In particular, multiple versions of the

same scene required for market research surveys directed the object insertion solution towards dynamic compositing. This, in turn, led to a 3D game engine as the choice of the environment for implementation, as only game engines provide the required real-time rendering tools. Given that only Unity was compatible with the asset production pipelines of my host companies, this was the game engine I used in my project. This entailed a need for the pipeline design to be tailored for Unity with its features but also flaws that would not be present if another game engine was used.

7.1.1 *Standard Videos*

There are many applications of game engines to real-time compositing with a video stream, mostly for augmented and mixed reality, previsualisation and virtual production. However, there is no mention in the literature of using game engines for traditional compositing with a pre-tracked camera. My method fills in this gap.

Chapter 3 contains a detailed description of the designed pipeline for dynamic mixed-reality compositing in Unity for standard videos. It takes a pre-tracked camera motion and a sparse scene reconstruction as input and produces a Unity scene with virtual objects inserted into a pre-recorded camera footage in real time. A vital part of the pipeline is a synchronisation mechanism that allows the event-driven virtual camera animation to be synchronised with the clock-driven recorded video footage. It was designed to overcome the issues, firstly with a known Unity bug in interpolating the imported animation curve, and secondly with different frame sampling policies for animations and videos. Without a reliable synchronisation of these two types of events, it would not be possible to perform compositing inside Unity.

In addition to the camera path, the input tracking data contains a sparse scene reconstruction. Reconstructed points are used to estimate a simple local scene geometry by fitting a plane to a subset of the points selected by the user. The plane is then used in two ways. Firstly, to guide graphics insertions by acting as a ground plane or a wall. A custom script snaps the object to the plane surface and aligns its rotation with plane directions making it easier to insert virtual objects in the correct place. Secondly, the plane is used as a component of differential rendering, implemented to provide image-based shadowing for greater realism of virtual objects.

At first, I tested only the synchronisation mechanism on a toy example of a synthetic scene, which helped me to find a working solution. Then, I tested the full dynamic compositing pipeline on video footage recorded by various standard cameras. The input tracking data was obtained from two different pieces of camera tracking software, and from an experimental hardware tracking system. In all cases, the synchronisation works correctly and the inserted virtual objects stay fixed in place when the camera moves. The Unity game engine environment also allows users to interact with the CG objects, e.g. move

and rotate them, change their textures or replace them with other objects in real-time, making the compositing dynamic.

7.1.2 360° Videos

Having explained the dynamic compositing in the Unity game engine for standard videos in Chapter 3, I focused on its extensions and modifications for 360° footage in Chapters 5 and 6.

In Chapter 5, I described the designed omnidirectional structure-from-motion algorithm. The algorithm combines partial solutions from literature to build the three main SfM modules: feature tracker, camera solver and bundle adjustment. Some of the solutions used were originally created for standard cameras or other types of omnidirectional cameras (usually catadioptric). They were all adjusted to work together for omnidirectional spherical cameras.

The *feature tracker* tracks features directly on equirectangular images, which are the most popular 360° image format, but there is no prior mention of this choice of the input format for tracking in literature. It is based on a standard KLT tracker with added custom features. Taking into account the non-standard image geometry, it treats an image as continuous along its horizontal axis, so the tracking does not stop at the left or right image edge.

The *camera solver* exploits the omnidirectional version of epipolar geometry to calculate the camera pose in every frame according to feature correspondences tracked between the frames. It also uses calculated camera poses and feature correspondences to reconstruct 3D points.

Hierarchical *bundle adjustment* refines the solution by minimising the reprojection error. Traditional reprojection error measured in pixels does not correspond with the highly non-linear projection of 3D space into an equirectangular image and was replaced by the tangential reprojection error in the spherical domain.

Chapter 6 contains the description of the designed complete pipeline for real-time inserting CG objects into 360° videos from a moving camera. So far, only subsets of the complete pipeline can be found in literature.

Omnidirectional SfM provides input data – the camera motion and reconstructed points – for the dynamic compositing part. The synchronisation between the camera animation and 360° video frames is analogous to this described for standard videos, except that the spherical content is displayed on the inside surface of a sphere rather than on a plane. This requires a modification of the previous implementation of differential rendering for shadowing and introducing an intermediary virtual standard camera. It acts as a projector and transfers a portion of a spherical image into a perspective view. Then, the image can be processed like a standard perspective one.

In the case of 360° videos, there is no need for acquiring separate environment maps for image-based lighting, as each frame already provides the required spherical geometry.

As pre-processing steps, I stabilise the video and apply the inverse tone mapping to the frames to convert them to HDR environment maps for a more accurate representation of lighting conditions in the scene. I also exploit the information about the position of each frame in space to sample frames from the video and create a set of local environment maps distributed along the camera path for spatially-varying reflections.

I tested my SfM algorithm on a set of synthetic data (a pointcloud) and synthetic videos (360° renderings of virtual scenes), and the whole pipeline on a synthetic video and various video sequences recorded with different models of spherical 360° cameras at different resolutions and frame rates. Structure from motion algorithm reproduces the 360° camera movement and reconstructs the sparse geometry of a scene with sufficient accuracy. It was compared to ground truth, where available, with satisfactory results (limitations and proposed improvements are described in Section 7.3 below). The complete object insertion pipeline produces mixed-reality omnidirectional scenes with dynamic shadows and reflections changing according to the position of CG objects when the user interacts with them.

7.2 A REAL-LIFE MARKET RESEARCH USE CASE

The remaining two research objectives are related to applications of the designed method to insert CG objects into real footage in the context of market research:

- O2: To test the method on a real-world market research use case.
- O3: To compare the existing virtual reality market research environments created by my industrial collaborators with the new mixed-reality ones created using my method. In particular, to test if the research study participants behave more naturally in mixed-reality environments than they do in purely CG ones or, at least if they behave in the same way in both types of environments.

These two research objectives intertwine as the output of the first one provides the input for the second one. The results referring to both are described in Chapter 4.

The real-world use case to test my method was inspired by the past market research survey designed by my host company in a fully CG environment. Initially, the goal was to re-create a part of that survey, a car journey to the petrol station and stopping at the pump. However, that was not entirely possible for various reasons. Firstly, the original CG scene comprised of a generic city with a petrol station with no reference to a real place. Secondly, the mixed-reality video was intended to be a proof of concept and reflect the capabilities of the new method, which encouraged augmenting it with as many CG objects as possible.

It was the first time a mixed-reality video was used to create dynamically changing video assets for market research surveys. This confirmed that the designed mixed-reality pipeline can be used for this purpose. In addition, my host company used it for pitching their services to potential clients, which increased the impact of the project.

In the next step, the produced mixed-reality video was compared with the original CG version of the petrol station scene. I designed and conducted a user study to evaluate three factors: perceptiveness score of each video, the feeling of presence and the general preference of the participants. Perceptiveness score indicates the number and type of elements that participants remember best (or remember at all) from the environment. This is useful for planning the location of hotspots in the video to make sure they are noticed, as their role is to influence customers' behaviour. The feeling of presence measures the degree of realism people associate with the environment, which translates directly into how natural they behave. General preference combines factors that participants cannot articulate but perceive subconsciously as "wrong" or "right". That makes them prefer one video over the other, but also influences their behaviour.

Despite the limitations of the user study, it resulted in a meaningful comparison of the two types of video assets and recommendations on how to improve the design of both. All three hypotheses were confirmed or partially confirmed. Firstly, the participants preferred the MR video, mainly for its greater realism. Secondly, they admitted that realism encourages more natural behaviour. Thirdly, the study proved that the MR video is equally flexible as the CG one in terms of content, but, unfortunately, the latter scored higher in the perceptiveness test. This drove my attention to the reason why this happened. Analysis of participants' comments led to the conclusion that people remember more from the scene if it contains fewer elements to remember or fewer distracting elements. That indicates that MR videos better model real environments, which was not considered in the original hypothesis.

The study confirmed that MR videos, which are easier to produce, can replace fully CG ones in market research surveys. It also revealed that standard videos, MR or CG, are not sufficiently immersive to provide the high feeling of presence. This directs the research towards the new type of immersive videos, spherical 360° videos, and their applications to creating market research video assets.

7.3 LIMITATIONS AND FUTURE WORK

There are some limitations to my pipeline for inserting CG objects into real videos, and it can be further enhanced by the following features.

7.3.1 *Research Challenges*

Automatic light estimation. Inserting a light to the virtual scene, which is done manually, could be replaced by more convenient to use automatic estimation of light position. Basic, but sufficiently accurate, light position estimation techniques utilise pixel intensities of an additional HDR map (for standard videos) or HDR versions of spherical video frames currently used for image-based lighting and reflections (for 360° videos).

Dense scene reconstruction. While a simple planar geometry reconstruction for guiding the object insertions and for shadowing is sufficient in most cases, it does not support more advanced interactions between the inserted objects and the real scene. More complex geometry reconstruction would not limit the insertions only to flat areas, would allow them to be occluded by real objects and improve the quality of shadowing. That requires utilising depth information of the scene, either from external depth sensors or from the dense reconstruction.

The leading augmented reality libraries understand the importance of introducing real-time depth estimation to the image. Currently, they go in two directions in its development. Google's ARCore [37] and its recently introduced ARCore Depth API aims at using non-specialised mobile devices, where "non-specialised" means a device equipped with a single RGB camera, without a depth sensor. Depth is estimated based on several views of the scene from different positions of the camera. Apple's ARKit [4], in turn, relies on the built-in depth sensor, TrueDepth Camera, although it uses depth data only for face recognition and people occlusion. Huawei's AR Engine [44] offers 3D scanning and gesture and body pose recognition also only for particular devices equipped with a depth sensor.

These are real-time solutions that create opportunities for future off-line systems. If the estimated depth information was stored with the corresponding RGB frame, it could be later re-used off-line. Not limited by real-time requirements for speed and efficiency, depth data may then undergo post-processing to increase its quality before it is used off-line. Post-processing steps include, for example, filling the holes, edge anti-aliasing or introducing temporal consistency.

Improved feature tracking for omnidirectional structure from motion. In the omnidirectional SfM, there is still room for improvement in the feature tracking solution. The standard KLT tracker, which runs directly on equirectangular images, proved to work sufficiently accurately for the most common horizontal camera movement. In this type of movement, tracked features move along the middle band of the equirectangular frame, the area with the smallest image distortion. However, if a feature moves out of this band towards the poles, which happens to most of the features during vertical camera movement, it becomes too distorted to be reliably tracked. This results in short tracking trajectories which reduce the accuracy and, in the extreme cases, break the final camera solution.

Even in the case of horizontal camera movement, with accurately tracked camera and reconstructed point cloud, there are usually very few points reconstructed on the ground. The reason is the same as above: ground features are located in the bottom part of an equirectangular image, and therefore, undergo a significant distortion, considerably changing from frame to frame. Arguably, these are the most important points for inserting CG objects into the scene, because they are used to form a ground plane for insertions. The fewer points the less accurate the estimated planar geometry position and angle become.

The proposed solution is to move tracking from the equirectangular image domain to spherical domain to eliminate the influence of non-uniform pixel distortion and, in result, to obtain longer and more reliable feature trajectories.

Testing the omnidirectional pipeline on a real-world market research use case. At this stage, I tested my pipeline only on a real-world market research use case involving standard videos. One of the findings of the user study I conducted to compare the output of the pipeline with fully computer-generated videos indicated the need for more immersive videos than the standard ones. Mixed-reality 360° videos have potential to introduce more immersion into research environments and are flexible enough to produce the required number of different versions of the same scene. Therefore, it is worthwhile to test the application of the omnidirectional pipeline to market research surveys and to conduct a user study similar to that for the standard videos.

7.3.2 Engineering Challenges

Interpolation between the reflection probes. The feature that requires cosmetic changes to improve its look is the way reflections change when a moving virtual reflective object crosses the boundary between the influence volumes of two neighbouring reflection probes. Unity's default interpolation between the reflection probes uses constant weights, which produces ghosting artefacts and was discarded. However, this makes the reflections switch between the two maps, which is not visually pleasant and appears artificial to the viewer. For a gradually changing reflection, one needs a custom interpolation between the two nearest reflection maps with weights that depend on the distance between the object and the corresponding reflection probes.

Loading video frames from a remote server. Currently, the final compositing step of the pipeline is implemented as a self-contained Unity project. All its components, including the recorded video frames, are located in the assets folder and are included in the project build file. This increases the project size and, in consequence, leads to longer loading time if the scene is loaded online. Moving video frames to a remote server and loading them in batches at runtime would dramatically reduce the project size and increase the pipeline's usability.

Full integration with the host companies' pipelines. The designed method has been less integrated than initially anticipated with my host companies' pipelines for creating market research surveys. It is understandable, as it should not be expected for the innovative tools to be immediately applied by a company to their projects. With a successful proof of concept mixed-reality video, the next step would be to complete the integration with the Scenario Manager. The Scenario Manager is a patent-pending system developed

by Dc-activ to control the combinations of market research survey versions shown to each study participant and to record associated participants' answers and interactions with the surveys.

7.4 MY REFLECTIONS ON CONDUCTING RESEARCH IN ACADEMIA AND IN INDUSTRY

Having been involved in an academic research project in an industrial environment, I had a chance to observe and compare different approaches to conducting research in both academia and industry. In my opinion, the primary differences lie in the definition of research, the duration of research projects and their expected outcome.

Industry tends to target short projects that bring immediate benefits. The main goal is to make a complete working system in the shortest possible time, so there is usually no time for in-depth theory research. Therefore, industry projects are generally based on already published research and off-the-shelf, sometimes also open source, software. They do not have to be technically correct if they produce the desired or close to desired output. Moreover, the industry does not prioritise research in its goals. If there is no designated research team, or if there is an abundance of deadlines and shortage of workforce, research will always be postponed.

Academia follows a different approach. In academia, projects may last for years and are not pursued with financial benefits in mind. The main goal is to push the boundaries of knowledge and, especially, to publish the results. To do so, the academic research project requires novelty. The solution may be designed only for a small subset of a general problem and leave more detailed applications and implementations to future work (continued by the same or another research team). Only big companies, such as Google, Microsoft or Adobe, with teams of research engineers designated to specific projects, can afford the academic approach to research. It allows them to implement their research and present it at top conferences at the same time.

Between these two worlds exists the Engineering Doctorate program. It consists of a three-year industrial placement that should strike a balance between in-depth academic research and publishing its results and a full implementation that the host company can incorporate in their everyday projects. However, typically, there is a usual few years gap between a paper publication and the application of its content to the industry. Therefore, it is very difficult for a doctoral researcher alone to complete a project within three years, both contributing to the research field and being useful for the host company. It is even more difficult if the company cannot spend a lot of resources (human or financial) to support the project. There is also a considerable issue with ownership rights to the project code, which belong to the host company. It prevents the code to be released to the public – a common practice in academic papers. The code is then unlikely to be included in other academic projects, which reduces the opportunities of collaboration and leads to increased isolation of the research.

On the other hand, the EngD program offers experience in the industry without a full commitment to an industrial job position. One can compare it to a “trial period”. It includes becoming familiar with external project management and meeting short-term deadlines, unusual in academia, which is focused on long-term deadlines and personal responsibility for time and project management. Working in industry also means applying research to real-world use cases, where formulating the problem precedes finding a solution, which, sadly, is not always applicable to academia.

To sum up my experience, pursuing an EngD requires even more perseverance and stamina than pursuing a standard PhD, but it better prepares the student for diversified career paths after graduation.

7.5 CONCLUDING REMARKS

In this dissertation, I aimed at addressing the main technical challenges associated with graphics insertions into real videos and their application to creating video assets for market research. In my work, I investigated both standard perspective videos and relatively new to the consumer market 360° spherical videos. I hope that, in the long term, the outcome of this project proves useful to improve future market research surveys, as well as it finds its use in more general real-time compositing applications. With the borderline between offline and real-time systems becoming increasingly blurry, and the constantly increasing rendering quality of game engines, further research in this direction is certainly worth pursuing.

BIBLIOGRAPHY

- [1] Sameer Agarwal, Keir Mierle, and Others. *Ceres Solver*. <http://ceres-solver.org>. 2010-2018.
- [2] Pablo Alcantarilla, Jesus Nuevo, and Adrien Bartoli. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces." *Proceedings of the British Machine Vision Conference 2013* (2013), pp. 13.1–13.11. DOI: [10.5244/C.27.13](https://doi.org/10.5244/C.27.13).
- [3] Robert Anderson, David Gallup, Jonathan T Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M Seitz. "Jump: Virtual Reality Video." *ACM Trans. Graph. Article* 3516.1312 (2016), pp. 978–1. DOI: [10.1145/2980179.2980257](https://doi.org/10.1145/2980179.2980257).
- [4] Apple. *ARKit | Apple Developer Documentation*. 2020. URL: <https://developer.apple.com/documentation/arkit>.
- [5] Ronald T. Azuma. "A Survey of Augmented Reality." *Presence: Teleoperators and Virtual Environments* 6.4 (1997), pp. 355–385.
- [6] Housseem Eddine Benseddik, Hicham Hadj - Abdelkader, Brahin Cherki, and Oualid Abdel Djekoune. "Binary Feature Descriptor for Omnidirectional Images Processing." *International Conference on Intelligent Information Processing, Security and Advanced Communication* December (2015). DOI: [10.1145/2816839.2816848](https://doi.org/10.1145/2816839.2816848).
- [7] Mehmet Ilker Berkman and Ecehan Akan. "Presence and Immersion in Virtual Reality." In: *Encyclopedia of Computer Graphics and Games*. Ed. by Newton Lee. Cham: Springer International Publishing, 2019, pp. 1–10. DOI: [10.1007/978-3-319-08234-9_162-1](https://doi.org/10.1007/978-3-319-08234-9_162-1). URL: https://doi.org/10.1007/978-3-319-08234-9_162-1 (visited on 09/01/2020).
- [8] Mark Billinghurst, Adrian Clark, and Gun Lee. "A Survey of Augmented Reality." *Foundations and Trends® in Human-Computer Interaction* 8.2-3 (2015), pp. 73–272. DOI: [10.1561/11000000049](https://doi.org/10.1561/11000000049).
- [9] Brandon Bray and Matt Zeller. *What is Mixed Reality?* 2018. URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality> (visited on 11/09/2019).
- [10] Erik Brynjolfsson, Yu Jeffrey Hu, and Mohammad S. Rahman. "Competing in the age of omnichannel retailing." *MIT Sloan Management Review* 54.4 (2013), pp. 1–7.
- [11] Dan A. Calian, Jean-François Lalonde, Paulo Gotardo, Tomas Simon, Iain Matthews, and Kenny Mitchell. "From Faces to Outdoor Light Probes." *ProcEG* 37.2 (May 2018), pp. 51–61. DOI: [10.1111/cgf.13341](https://doi.org/10.1111/cgf.13341).

- [12] Thomas P. Caudell and David W. Mizell. "Augmented reality: an application of heads-up display technology to manual manufacturing processes." *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences* (1992), 659–669 vol.2. DOI: [10.1109/HICSS.1992.183317](https://doi.org/10.1109/HICSS.1992.183317).
- [13] Peng Chang and Martial Hebert. "Omni-directional structure from motion." *Proceedings - IEEE Workshop on Omnidirectional Vision, OMNIVIS 2000* (2000), pp. 127–133. DOI: [10.1109/OMNIVIS.2000.853819](https://doi.org/10.1109/OMNIVIS.2000.853819).
- [14] Oliver Cossairt, Shree K. Nayar, and Ravi Ramamoorthi. "Light field transfer: global illumination between real and synthetic objects." *ProcSIGGRAPH* 27.3 (Aug. 2008), 57:1–6. DOI: [10.1145/1360612.1360656](https://doi.org/10.1145/1360612.1360656).
- [15] Javier Cruz-Mota, Iva Bogdanova, Benoît Paquier, Michel Bierlaire, and Jean-Philippe Thiran. "Scale Invariant Feature Transform on the Sphere: Theory and Applications." *International Journal of Computer Vision* 98.2 (2012), pp. 217–241. DOI: [10.1007/s11263-011-0505-4](https://doi.org/10.1007/s11263-011-0505-4).
- [16] Ross Cutler. *Omni-directional camera design for video conferencing*. US Patent 7,298,392. 2007.
- [17] Erik B. Dam, Martin Koch, and Martin Lillholm. *Quaternions, Interpolation and Animation*. Tech. rep. DIKU-TR-98/5. Datalogisk Institut, Københavns Universitet, July 1998.
- [18] Paul Debevec. "Rendering Synthetic Objects into Real Scenes : Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography." In *Computer Graphics Proceedings, Annual Conference Series (Proc. ACM SIGGRAPH '98 Proceeding)* (1998), pp. 189–198. DOI: [10.1145/280814.280864](https://doi.org/10.1145/280814.280864).
- [19] Paul Debevec. "Image-based lighting." *CGA* 22.2 (2002), pp. 26–34. DOI: [10.1109/38.988744](https://doi.org/10.1109/38.988744).
- [20] Paul Debevec and Jitendra Malik. "Recovering High Dynamic Range Radiance Maps from Photographs." In: *SIGGRAPH*. Aug. 1997, pp. 369–378.
- [21] Paul Debevec and Chris Tchou. *HDR Shop v1.0.3*. URL: www.hdrshop.com.
- [22] Tom Duff. "Compositing 3-D rendered images." *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* 19.3 (1985), pp. 41–44. DOI: [10.1145/325334.325174](https://doi.org/10.1145/325334.325174).
- [23] David Eberly. *Least Squares Fitting of Data by Linear or Quadratic Structures*.
- [24] David Eberly. *Conversion of Left-Handed Coordinates to Right-Handed Coordinates*. 2008. URL: <http://answers.unity3d.com/storage/temp/12048-lefthandedtorighthanded.pdf> (visited on 05/05/2017).

- [25] 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* 36.6 (2017), pp. 1–15. DOI: [10.1145/3130800.3130816](https://doi.org/10.1145/3130800.3130816). arXiv: [1710.07480](https://arxiv.org/abs/1710.07480).
- [26] Farshad Einabadi and Oliver Grau. "Discrete Light Source Estimation from Light Probes for Photorealistic Rendering." *British Machine Vision Conference (BMVC)* (2015), pp. 1–10.
- [27] Yuki Endo, Yoshihiro Kanamori, and Jun Mitani. "Deep Reverse Tone Mapping." *ACM Transactions on Graphics (Proc. of SIGGRAPH ASIA 2017)* 36.6 (Nov. 2017).
- [28] Epic Games. *Unreal Engine 4*. URL: www.unrealengine.com.
- [29] Facebook. *Next-generation video encoding techniques for 360 video and VR*. URL: <https://code.fb.com/virtual-reality/next-generation-video-encoding-techniques-for-360-video-and-vr/>.
- [30] Foundry. *Cara VR*. URL: <https://www.foundry.com/products/cara-vr>.
- [31] Jun Fujiki, Akihiko Torii, and Shotaro Akaho. "Epipolar Geometry Via Rectification of Spherical Images." *Lecture Notes in Computer Science* 4418 (2007), pp. 461–471. DOI: [10.1007/978-3-540-71457-6_42](https://doi.org/10.1007/978-3-540-71457-6_42).
- [32] Thomas A Furness. *The application of helmet-mounted displays to airborne reconnaissance and weapon delivery. Technical Report TR-69-241*. Wright-Patterson Air Force Base, OH: U.S. Air Force Avionics Laboratory, 1969.
- [33] Thomas A Furness. "The Super Cockpit and its Human Factors Challenges." *Proceedings of the Human Factors Society Annual Meeting* 30.1 (1986), pp. 48–52. DOI: [10.1177/154193128603000112](https://doi.org/10.1177/154193128603000112).
- [34] 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." *ProcSIGASIA* 36.6 (Nov. 2017), 176:1–14. DOI: [10.1145/3130800.3130891](https://doi.org/10.1145/3130800.3130891).
- [35] Google. *Bringing pixels front and center in VR video*. URL: <https://blog.google/products/google-ar-vr/bringing-pixels-front-and-center-vr-video/>.
- [36] Google. *[discontinued]ATAP Project Tango - Google*. 2014. URL: <http://www.google.com/atap/projecttango/>.
- [37] Google. *ARCore - Google Developers Documentation*. 2020. URL: <https://developers.google.com/ar>.
- [38] Lukas Gruber, Thomas Richter-Trummer, and Dieter Schmalstieg. "Real-time photometric registration from arbitrary geometry." In: *ISMAR*. Nov. 2012, pp. 119–128. DOI: [10.1109/ISMAR.2012.6402548](https://doi.org/10.1109/ISMAR.2012.6402548).

- [39] Richard I. Hartley and Peter Sturm. "Triangulation." *CVIU* 68.2 (1997), pp. 146–157. DOI: [10.1006/cviu.1997.0547](https://doi.org/10.1006/cviu.1997.0547).
- [40] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.
- [41] Tim Hilken, Jonas Heller, Mathew Chylinski, Debbie Isobel Keeling, Dominik Mahr, and Ko de Ruyter. "Making omnichannel an augmented reality: the current and future state of the art." *Journal of Research in Interactive Marketing* 12.4 (2018), pp. 509–523. DOI: [10.1108/JRIM-01-2018-0023](https://doi.org/10.1108/JRIM-01-2018-0023).
- [42] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. "Deep Outdoor Illumination Estimation." In: *CVPR*. July 2017, pp. 2373–2382. DOI: [10.1109/CVPR.2017.255](https://doi.org/10.1109/CVPR.2017.255). URL: <http://vision.gel.ulaval.ca/~jflalonde/projects/deepOutdoorLight/>.
- [43] Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. "6-DOF VR videos with a single 360-camera." *Proceedings - IEEE Virtual Reality* (2017), pp. 37–44. DOI: [10.1109/VR.2017.7892229](https://doi.org/10.1109/VR.2017.7892229).
- [44] Huawei. *Huawei AR Engine Guides*. 2020. URL: <https://developer.huawei.com/consumer/en/doc/development/HUAWEI-AR-Guides/20301>.
- [45] Sunghoon Im, Hyowon Ha, François Rameau, Hae-Gon Jeon, Gyeongmin Choe, and In So Kweon. "All-Around Depth from Small Motion with a Spherical Panoramic Camera." In: *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Vol. 9908. Springer International Publishing, 2016, pp. 156–172. DOI: [10.1007/978-3-319-46487-9_10](https://doi.org/10.1007/978-3-319-46487-9_10). eprint: [1311.2901](https://doi.org/1311.2901).
- [46] Thomas Iorns and Taehyun Rhee. "Real-Time Image Based Lighting for 360-Degree Panoramic Video." In: *PSIVT Workshops*. 2015, pp. 139–151. DOI: [10.1007/978-3-319-30285-0_12](https://doi.org/10.1007/978-3-319-30285-0_12).
- [47] Thomas Iorns and Taehyun Rhee. "Real-time image based lighting for 360-degree panoramic video." In: *Lecture Notes in Computer Science*. Vol. 9555. Springer Verlag, 2016, pp. 139–151.
- [48] Darko Jurić. *Accord.NET Extensions*. <https://github.com/dajuric/accord-net-extensions>. 2015.
- [49] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Forward-backward error: Automatic detection of tracking failures." *Proceedings - International Conference on Pattern Recognition* (2010), pp. 2756–2759. DOI: [10.1109/ICPR.2010.675](https://doi.org/10.1109/ICPR.2010.675).
- [50] Sing Bing Kang and Richard Szeliski. "3-D scene data recovery using omnidirectional multibaseline stereo." *International Journal of Computer Vision* 25.2 (1997), pp. 167–183. DOI: [10.1023/A:1007971901577](https://doi.org/10.1023/A:1007971901577).

- [51] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. "Rendering synthetic objects into legacy photographs." *ProcSIGASIA* 30.6 (Dec. 2011), 157:1–12. DOI: [10.1145/2070781.2024191](https://doi.org/10.1145/2070781.2024191). URL: <http://kevinkarsch.com/publications/sa11.html>.
- [52] Patrick Keating, ed. *Cinematography*. Behind the silver screen. I.B. Tauris & Company Limited, 2014.
- [53] Sebastian B. Knorr and Daniel Kurz. "Real-time illumination estimation from faces for coherent rendering." In: *ISMAR*. Sept. 2014, pp. 113–122. DOI: [10.1109/ISMAR.2014.6948416](https://doi.org/10.1109/ISMAR.2014.6948416).
- [54] Robert Konrad, Donald G. Dansereau, Aniq Masood, and Gordon Wetzstein. "SpinVR: Towards Live-streaming 3D Virtual Reality Video." *ACM Trans. Graph.* 36.6 (Nov. 2017), 209:1–209:12. DOI: [10.1145/3130800.3130836](https://doi.org/10.1145/3130800.3130836).
- [55] Joel Kronander, Francesco Banterle, Andrew Gardner, Ehsan Miandji, and Jonas Unger. "Photorealistic rendering of mixed reality scenes." *Computer Graphics Forum* 34.2 (2015), pp. 643–665. DOI: [10.1111/cgf.12591](https://doi.org/10.1111/cgf.12591).
- [56] Myron W. Krueger. *Artificial Reality II*. Polar Research. Addison-Wesley, 1991.
- [57] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. "Estimating the Natural Illumination Conditions from a Single Outdoor Image." *IJCV* 98.2 (2012), pp. 123–145. DOI: [10.1007/s11263-011-0501-8](https://doi.org/10.1007/s11263-011-0501-8).
- [58] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints." In: *2011 International Conference on Computer Vision*. 2011, pp. 2548–2555. DOI: [10.1109/ICCV.2011.6126542](https://doi.org/10.1109/ICCV.2011.6126542).
- [59] Lightcraft Technology. *Previzion, Real-Time VFX*. 2004. URL: <http://www.lightcrafttech.com> (visited on 07/24/2017).
- [60] Yu-Lun Liu, Wei-Sheng Lai, Yu-Sheng Chen, Yi-Lung Kao, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. "Single-Image HDR Reconstruction by Learning to Reverse the Camera Pipeline." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [61] Thiago Lopes Trugillo Da Silveira and Claudio Rosito Jung. "Evaluation of Keypoint Extraction and Matching for Pose Estimation Using Pairs of Spherical Images." In: *Proceedings - 30th Conference on Graphics, Patterns and Images, SIBGRAPI 2017*. 2017, pp. 374–381. DOI: [10.1109/SIBGRAPI.2017.56](https://doi.org/10.1109/SIBGRAPI.2017.56).
- [62] Jorge Lopez-Moreno, Elena Garces, Sunil Hadap, Erik Reinhard, and Diego Gutierrez. "Multiple Light Source Estimation in a Single Image." *Computer Graphics Forum* 32.8 (Dec. 2013), pp. 170–182. DOI: [10.1111/cgf.12195](https://doi.org/10.1111/cgf.12195).
- [63] Bruce D. Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." In: *IJCAI*. 1981.

- [64] Chuiwen Ma, Liang Shi, Hanlu Huang, and Mengyuan Yan. "3D Reconstruction from Full-view Fisheye Camera" (2015). arXiv: [1506.06273](https://arxiv.org/abs/1506.06273). URL: <http://arxiv.org/abs/1506.06273>.
- [65] Steve Mann. *Mediated Reality*. Tech. rep. University of Toronto, 1994.
- [66] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. "Pose Estimation for Augmented Reality: A Hands-On Survey." *TVCG* 22.12 (Dec. 2016), pp. 2633–2651. DOI: [10.1109/TVCG.2015.2513408](https://doi.org/10.1109/TVCG.2015.2513408).
- [67] Demetris Marnierides, Thomas Bashford-Rogers, Jonathan Hatchett, and Kurt Debattista. "ExpandNet: A Deep Convolutional Neural Network for High Dynamic Range Expansion from Low Dynamic Range Content." *Computer Graphics Forum* 37 (2018), pp. 37–49.
- [68] Mettle. *SkyBox Studio V2*. URL: <https://www.mettle.com/product/skybox-studio-v2/>.
- [69] Nick Michiels, Lode Jorissen, Jeroen Put, and Philippe Bekaert. "Interactive augmented omnidirectional video with realistic lighting." In: *AVR 2014. LNCS*. Ed. by Lucio Tommaso De Paolis and Antonio Mongelli. Vol. 8853. Heidelberg: Springer, 2014, pp. 247–263. DOI: [10.1007/978-3-319-13969-2_19](https://doi.org/10.1007/978-3-319-13969-2_19).
- [70] Paul Milgram and Fumio Kishimo. "A taxonomy of mixed reality." *IEICE Transactions on Information and Systems* 77.12 (1994), pp. 1321–1329.
- [71] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. "Augmented reality: a class of displays on the reality-virtuality continuum." *Proceedings of Telemanipulator and Telepresence Technologies* 2351 (1994), pp. 282–292. DOI: [10.1117/12.197321](https://doi.org/10.1117/12.197321).
- [72] Shree K. Nayar. "Catadioptric Omnidirectional Camera." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1997, pp. 482–488.
- [73] Ncam Technologies Ltd. *Ncam, AR/VR Realtime Camera Tracking*. 2012. URL: <http://www.ncam-tech.com> (visited on 07/24/2017).
- [74] Ernest North and Retha De Vos. "The use of conjoint analysis to determine consumer buying preferences: A literature review." *Journal of Family Ecology and Consumer Sciences* 30.1 (2010), pp. 32–39.
- [75] Lesley Northam, Joe Istead, and Craig S. Kaplan. "RTFX: On-set previs with UnrealEngine3." *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6972 LNCS (2011), pp. 432–435.
- [76] Alain Pagani and Didier Stricker. "Structure from Motion using full spherical panoramic cameras." *Computer Vision Workshops* (2011), pp. 375–382. DOI: [10.1109/ICCVW.2011.6130266](https://doi.org/10.1109/ICCVW.2011.6130266).

- [77] Sarthak Pathak, Alessandro Moro, Hiromitsu Fujii, Atsushi Yamashita, and Hajime Asama. "3D reconstruction of structures using spherical cameras with small motion." *International Conference on Control, Automation and Systems Iccas* (2017), pp. 117–122. DOI: [10.1109/ICCAS.2016.7832307](https://doi.org/10.1109/ICCAS.2016.7832307).
- [78] Thomas Porter and Tom Duff. "Compositing digital images." *ACM SIGGRAPH Computer Graphics* 18.3 (1984), pp. 253–259. DOI: [10.1145/964965.808606](https://doi.org/10.1145/964965.808606).
- [79] Richard J. Radke. *Computer Vision for Visual Effects*. New York, NY, USA: Cambridge University Press, 2013.
- [80] Taehyun Rhee, Lohit Petikam, Benjamin Allen, and Andrew Chalmers. "MR360: Mixed Reality Rendering for 360° Panoramic Videos." *IEEE Transactions on Visualization and Computer Graphics* 23.4 (2017), pp. 1302–1311.
- [81] Thomas Richter-Trummer, Denis Kalkofen, Jinwoo Park, and Dieter Schmalstieg. "Instant Mixed Reality Lighting from Casual Scanning." In: *ISMAR*. 2016, pp. 27–36. DOI: [10.1109/ISMAR.2016.18](https://doi.org/10.1109/ISMAR.2016.18).
- [82] Marcel Santana Santos, Tsang Ing Ren, and Nima Khademi Kalantari. "Single Image HDR Reconstruction Using a CNN with Masked Features and Perceptual Loss." *ACM Trans. Graph.* 39.4 (July 2020). DOI: [10.1145/3386569.3392403](https://doi.org/10.1145/3386569.3392403).
- [83] Davide Scaramuzza. "Omnidirectional Camera." In: *Computer Vision: A Reference Guide*. Ed. by Katsushi Ikeuchi. Boston, MA: Springer US, 2014, pp. 552–560.
- [84] Davide Scaramuzza and Roland Siegwart. "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles." *IEEE Transactions on Robotics* 24.5 (2008), pp. 1015–1026. DOI: [10.1109/TRO.2008.2004490](https://doi.org/10.1109/TRO.2008.2004490).
- [85] Dieter Schmalstieg and Tobias Höllerer. *Augmented Reality: Principles and Practice*. Addison-Wesley usability and HCI series. Addison Wesley Professional, 2015.
- [86] Christopher Schroers, Jean-Charles Bazin, and Alexander Sorkine-Hornung. "An Omnistereoscopic Video Pipeline for Capture and Display of Real-World VR." *ACM Trans. Graph.* 37.3 (Aug. 2018), 37:1–37:13. DOI: [10.1145/3225150](https://doi.org/10.1145/3225150).
- [87] Jianbo Shi and Carlo Tomasi. "Good Features to Track." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Seattle, WA, 1994, pp. 593–600. DOI: [10.1109/CVPR.1994.323794](https://doi.org/10.1109/CVPR.1994.323794).
- [88] Mel Slater. "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments." *Philosophical Transactions of the Royal Society B: Biological Sciences* 364.1535 (2009), pp. 3549–3557. DOI: [10.1098/rstb.2009.0138](https://doi.org/10.1098/rstb.2009.0138).
- [89] Mel Slater. "Immersion and the illusion of presence in virtual reality." *British Journal of Psychology* 109.3 (2018), pp. 431–433. DOI: [10.1111/bjop.12305](https://doi.org/10.1111/bjop.12305).

- [90] Mel Slater, Amela Sadagic, Martin Usoh, and Ralph Schroeder. "Small-Group Behavior in a Virtual and Real Environment: A Comparative Study." *Presence: Teleoper. Virtual Environ.* 9.1 (Feb. 2000), pp. 37–51. DOI: [10.1162/105474600566600](https://doi.org/10.1162/105474600566600).
- [91] Mel Slater, Pankaj Khanna, Jesper Mortensen, and Insu Yu. "Visual Realism Enhances Realistic Response in an Immersive Virtual Environment." *IEEE Computer Graphics and Applications* 29.3 (2009), pp. 76–84. DOI: [10.1109/MCG.2009.55](https://doi.org/10.1109/MCG.2009.55).
- [92] Alvy Ray Smith. "Painting Tutorial Notes." *SIGGRAPH '79 Course on Computer Animation Techniques* (1979).
- [93] SolidAnim. *SolidTrack, The Innovative Markerless Camera Tracking System*. 2014. URL: <http://www.solid-track.com> (visited on 09/03/2016).
- [94] Maximilian Speicher, Brian D. Hall, and Michael Nebeling. "What is Mixed Reality?" In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland UK: ACM, 2019, 537:1–537:15.
- [95] Julian Straub et al. "The Replica Dataset: A Digital Replica of Indoor Spaces." 2019. URL: <https://github.com/facebookresearch/Replica-Dataset>.
- [96] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems." In: 2012. URL: <https://vision.in.tum.de/data/datasets/rgbd-dataset>.
- [97] Ivan E. Sutherland. "The ultimate display." *Proceedings of the Congress of the International Federation of Information Processing (IFIP)* (1965), pp. 506–508. DOI: [10.1109/MC.2005.274](https://doi.org/10.1109/MC.2005.274).
- [98] Ivan E. Sutherland. "A Head-Mounted Three Dimensional Display." In: *Proceedings of AFIPS 68*. Vol. 33. pt 1. 1968, pp. 757–764. DOI: [10.1145/1476589.1476686](https://doi.org/10.1145/1476589.1476686).
- [99] Joanna Tarko and Christian Richardt. "Environment Reflections with 360° Videos using Omnidirectional Structure from Motion." In: *CVMP Short Papers*. Dec. 2018.
- [100] Joanna Tarko, Christian Richardt, and Peter Hall. "Dynamic Mixed-Reality Compositing with Unity." In: *CVMP Short Papers*. Dec. 2017.
- [101] Joanna Tarko, James Tompkin, and Christian Richardt. "OmniMR: Omnidirectional Mixed Reality with Spatially-Varying Environment Reflections from Moving 360° Video Cameras." In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, pp. 1177–1178. DOI: [10.1109/VR.2019.8798067](https://doi.org/10.1109/VR.2019.8798067).
- [102] Joanna Tarko, James Tompkin, and Christian Richardt. "Real-time Virtual Object Insertion for Moving 360° Videos." In: *Proceedings of the 17th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '19. Brisbane, QLD, Australia: ACM, 2019. DOI: [10.1145/3359997.3365708](https://doi.org/10.1145/3359997.3365708).
- [103] The Pixel Farm. *Spherical Tracking Toolset for PFTrack*. URL: <https://www.thepixelfarm.co.uk/pftrack/>.

- [104] Carlo Tomasi. "Detection and Tracking of Point Features." *School of Computer Science, Carnegie Mellon Univ.* 91.April (1991), pp. 1–22. DOI: [10.1016/S0031-3203\(03\)00234-6](https://doi.org/10.1016/S0031-3203(03)00234-6).
- [105] Akihiko Torii, Atsushi Imiya, and Naoya Ohnishi. "Two-and three-view geometry for spherical cameras." *Proc. of the Sixth Workshop on Omnidirectional Vision, Camera Networks and Non- classical Cameras* 105 (2005), pp. 29–34.
- [106] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew Fitzgibbon. "Bundle Adjustment – A Modern Synthesis." In: *ICCV*. 2000, pp. 298–372. DOI: [10.1007/3-540-44480-7_21](https://doi.org/10.1007/3-540-44480-7_21).
- [107] Joji Tsuruga and Eric Renaud-Houde. "The Human Race." In: *ACM SIGGRAPH 2017 Real Time Live! SIGGRAPH '17*. Los Angeles, California: ACM, 2017, pp. 26–26.
- [108] Iwan Ulrich and Illah Nourbakhsh. "Appearance-based place recognition for topological localization." *Proceedings - IEEE International Conference on Robotics and Automation* 2.April (2000), pp. 1023–1029.
- [109] Unity Technologies. *Unity*. URL: <http://unity3d.com>.
- [110] Martin Usoh, Ernest Catena, Sima Arman, and Mel Slater. "Using Presence Questionnaires in Reality." *Presence: Teleoper. Virtual Environ.* 9.5 (Oct. 2000), pp. 497–503. DOI: [10.1162/105474600566989](https://doi.org/10.1162/105474600566989).
- [111] Bruce A Wallace. "Merging and Transformation of Raster Images for Cartoon Animation." *Computer Graphics* 15.3 (1981), pp. 253–262.
- [112] Tuanfeng Y. Wang, Tobias Ritschel, and Niloy J. Mitra. "Joint Material and Illumination Estimation from Photo Sets in the Wild." In: *Proc3DV*. Sept. 2018, pp. 22–31. DOI: [10.1109/3DV.2018.00014](https://doi.org/10.1109/3DV.2018.00014).
- [113] Roger Watson and Helen Rappaport. *Capturing the Light*. Pan Books, 2014.
- [114] Bob G. Witmer and Michael J. Singer. "Measuring presence in virtual environments: A presence questionnaire." *Presence: Teleoperators and Virtual Environments* 7.3 (1998), pp. 225–240. DOI: [10.1162/105474698565686](https://doi.org/10.1162/105474698565686).
- [115] Paul R. Wolf and Bon A. Dewitt. *Elements of Photogrammetry: With Applications in GIS*. McGraw-Hill, 2000.
- [116] Wai Wong, Joanne Liew, Loo Chu Kiong, and Wei Kin Wong. "Omnidirectional Surveillance System for Digital Home Security." In: *Apr.* 2009, pp. 8–12. DOI: [10.1109/ICSAP.2009.13](https://doi.org/10.1109/ICSAP.2009.13).
- [117] S. Wright. *Compositing Visual Effects: Essentials for the Aspiring Artist*. Taylor & Francis, 2012.
- [118] Edward Zhang, Michael F. Cohen, and Brian Curless. "Emptying, Refurnishing, and Relighting Indoor Spaces." *ProcSIGASIA* 35.6 (Nov. 2016), 174:1–14. DOI: [10.1145/2980179.2982432](https://doi.org/10.1145/2980179.2982432).

- [119] Jinsong Zhang and Jean-François Lalonde. "Learning High Dynamic Range from Outdoor Panoramas." In: *ICCV*. 2017, pp. 4529–4538. doi: [10.1109/ICCV.2017.484](https://doi.org/10.1109/ICCV.2017.484).
- [120] Qiang Zhao, Wei Feng, Liang Wan, and Jiawan Zhang. "SPHORB: A Fast and Robust Binary Feature on the Sphere." *International Journal of Computer Vision* 113.2 (2015), pp. 143–159. doi: [10.1007/s11263-014-0787-4](https://doi.org/10.1007/s11263-014-0787-4).

Appendices



SCRIPTS FOR DYNAMIC COMPOSITING

A.1 SCRIPT FOR EXPORTING CAMERA ANIMATION FROM 3DS MAX TO UNITY

```

1  -----pre-declared variables-----
OPENG_T_MAXWORLD = Matrix3 [1, 0, 0] [0, 0, -1] [0, 1, 0] [0, 0, 0]
RIGHT_T_LEFT = Matrix3 [1, 0, 0] [0, 1, 0] [0, 0, -1] [0, 0, 0]

SCALE_FACTOR = 1 --default scale factor (scene in meters)
6  if (units.MetricType == #Centimeters) do (SCALE_FACTOR = 0.01)

-----functions-----

function maxToUnityConversion quatMax posCam =
11 (
    quatMaxCon = conjugate quatMax --because Unity uses left-handed rotation
    rotMatrixMax = quatMaxCon as Matrix3
    rotMatrixOGL = OPENG_T_MAXWORLD * rotMatrixMax
    rotMatrixUnity = RIGHT_T_LEFT * rotMatrixOGL * RIGHT_T_LEFT
16
    posOGL = posCam*OPENG_T_MAXWORLD --because Point3 is 1x3, not 3x1
    posUnity = posOGL * RIGHT_T_LEFT
    rotMatrixUnity[4] = posUnity
21
    rotMatrixUnity
)

-----main program-----

26 filePath = getSaveFileName()

if (filepath != undefined) do
(
    fStream = createFile filepath
    cam = selection[1]
31    cam.fovType = 2 --vertical FOV
    format "%\n" cam.curFOV to:fStream
    try
    (
36        for i=0 to (animationRange.end - 1) do
        (
            at time i
            (
                transformUnity = maxToUnityConversion cam.rotation cam.position
                camTranslation = transformUnity.translationpart
41                camRotation = transformUnity.rotationpart

                format "% % % % % % % %\n" (((i as integer)/ticksperframe)+1) (camTranslation.x *
                SCALE_FACTOR) (camTranslation.y * SCALE_FACTOR) (camTranslation.z * SCALE_FACTOR)
                camRotation.x camRotation.y camRotation.z camRotation.w to:fStream
                --set first frame count to 1

```

```
46     )
    )

    at time animationRange.end --to get rid of an empty line at the end of the file
    (
51         transformUnity = maxToUnityConversion cam.rotation cam.position
        camTranslation = transformUnity.translationpart
        camRotation = transformUnity.rotationpart

        format "% % % % % % % %" (((animationRange.end as integer)/ticksperframe)+1) (
        camTranslation.x * SCALE_FACTOR) (camTranslation.y * SCALE_FACTOR) (camTranslation.z *
        SCALE_FACTOR) camRotation.x camRotation.y camRotation.z camRotation.w to:fStream
56     )

    close fStream
    )
catch
61 (
    close fStream
    )
)
```

A.2 SYNCHRONISATION UNITY SCRIPT FOR STANDARD VIDEOS

```

1  using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using System.Text;
    using System.IO;
6  using System.Text.RegularExpressions;

    public class AnimateCameraCustom : MonoBehaviour //Run from Camera object
    {
        public TextAsset cameraMovement;
11     public string framePrefix;
        public int frameOffset = 0;
        public GameObject plane; //ground plane
        public GameObject finalComposite; //image plane with the final composite

16     List<string[]> cameraData = new List<string[]>();
        Camera cam;
        long frame = 2;
        Renderer rend;
        Renderer planeRend;
21     Renderer finalCompositeRend;
        Transform imagePlane;
        bool isPaused = false;
        bool playForward = true;
        bool nextFrameFlag = false;

26     void Start()
        {
            imagePlane = this.transform.Find("ImagePlane");
            cam = GetComponent<Camera>();
31     rend = imagePlane.GetComponent<Renderer>();
            planeRend = plane.GetComponent<Renderer>();
            finalCompositeRend = finalComposite.GetComponent<Renderer>();

            string allText = cameraMovement.text;
36     string[] textLines = Regex.Split(allText, "\n");

            for (int i = 0; i < textLines.Length; i++)
            {
                string[] line = Regex.Split(textLines[i], " ");
41     cameraData.Add(line);
            }

            cam.fieldOfView = float.Parse(cameraData[0][0]);
        }

46     void Update()

```

```

{
    if (Input.GetKey(KeyCode.Space)) isPaused = !isPaused;
    if (Input.GetKey(KeyCode.PageUp))
51     {
        playForward = false;
        nextFrameFlag = true;
    }
    if (Input.GetKey(KeyCode.PageDown))
56     {
        playForward = true;
        nextFrameFlag = true;
    }
}

61 void FixedUpdate()
{
    if (!isPaused)
    {
66         if (frame < cameraData.Count - 2) Play(frame);
        if (playForward && frame < cameraData.Count - 2) frame++;
        else
        {
            if (frame > 2) frame--;
71         }
    }
    else
    {
        //pause mode - playing frame by frame is possible
76         if(nextFrameFlag)
        {
            if (frame < cameraData.Count - 2) Play(frame);
            if (playForward && frame < cameraData.Count - 2) frame++;
            else
81             {
                if (frame > 2) frame--;
            }

            nextFrameFlag = false;
86         }
    }
}

void Play(long frame)
91 {
    string textureName = framePrefix + (frame + frameOffset).ToString("000");
    rend.material.mainTexture = Resources.Load<Texture2D>(textureName);
    planeRend.material.SetTexture("_Detail", Resources.Load<Texture2D>(textureName));

96    cam.transform.position = new Vector3(float.Parse(cameraData[(int)frame][1]),
                                           float.Parse(cameraData[(int)frame][2]),

```

```
101         float.Parse(cameraData[(int)frame][3]));  
        cam.transform.rotation = new Quaternion(-float.Parse(cameraData[(int)frame][4]),  
        float.Parse(cameraData[(int)frame][6]),  
        -float.Parse(cameraData[(int)frame][5]),  
        float.Parse(cameraData[(int)frame][7]));  
    }  
}
```

A.3 SYNCHRONISATION UNITY SCRIPT FOR 360° VIDEOS WITH REFLECTION PROBES

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Text.RegularExpressions;
5
public class CreateReflectionProbes : MonoBehaviour {

    //Run from Sphere object
    public int camPathSegs = 1;
    10 public TextAsset cameraMovement;
    public string framePrefix;
    public string hdrFramePrefix;
    public int frameOffset = 0; //number of the real first frame (the smallest number in
        frames count)
    public float fps = 29.97f;

    15 List<string[]> cameraData = new List<string[]>(); //each item is camera pose
    List<int> reflectionProbesFrames = new List<int>(); //list of frames that make a
        reflection probe
    List<Vector3> reflectionProbesPositions = new List<Vector3>();
    Skybox skyboxCam;

    20 long frame = 0;
    bool isPaused = true;
    bool playForward = true;
    bool nextFrameFlag = false;

    25 void Start () {

        LoadCameraData();
        CreateReflectionProbes();

    30

        //set interval for constant frame rate
        Time.fixedDeltaTime = 1 / fps;
        Time.maximumDeltaTime = 1 / fps;

    35 this.transform.position = new Vector3(float.Parse(cameraData[0][1]), float.Parse(
        cameraData[0][2]), float.Parse(cameraData[0][3]));
        //if there is also rotation in the data file:
        if (cameraData[0].Length > 4)
        {
            40 this.transform.rotation = new Quaternion(-float.Parse(cameraData[(int)frame][4]),
                float.Parse(cameraData[(int)frame][6]),
                -float.Parse(cameraData[(int)frame][5]),
                float.Parse(cameraData[(int)frame][7]));
        }
    }

```



```

45     GameObject[] allObjects = FindObjectsOfType<GameObject>();
        foreach (GameObject go in allObjects)
        {
            MeshRenderer mr = go.GetComponent<MeshRenderer>();
            if(mr != null) //e.g. reflection probes don't have Mesh Renderer but they count as
objects in the scene and are listed
50             mr.reflectionProbeUsage = UnityEngine.Rendering.ReflectionProbeUsage.Simple; //set
all reflections to reflection probe only, without probes blending, without skybox
        }
    }

    void Update ()
55    {
        if (Input.GetKey(KeyCode.Space)) isPaused = !isPaused;
        if (Input.GetKey(KeyCode.PageUp))
        {
            playForward = false;
            nextFrameFlag = true;
60        }
        if (Input.GetKey(KeyCode.PageDown))
        {
            playForward = true;
            nextFrameFlag = true;
65        }
    }

    void FixedUpdate()
70    {
        if (!isPaused)
        {
            if (frame < cameraData.Count - 1) Play(frame);

            if (playForward && frame < cameraData.Count - 1) frame++;
75            else
            {
                if (frame > 1) frame--;
            }
        }
80    }
    else
    {
        //pause mode - playing frame by frame is possible
        if (nextFrameFlag)
85        {
            if (frame < cameraData.Count - 1) Play(frame);

            if (playForward && frame < cameraData.Count - 1) frame++;
            else
90            {
                if (frame > 1) frame--;
            }
        }
    }

```

```

        nextFrameFlag = false;
    }
}
95
}

void Play(long frame)
{
100
    string textureName = framePrefix + (frame + frameOffset).ToString("000");

    //display frame inside a sphere
    this.GetComponent<Renderer>().material.mainTexture = Resources.Load<Texture2D>("
OmnidirectionalImages/" + textureName);

105
    //update global skybox with HDR version of current frame
    string textureNameHDR = "OmnidirectionalImagesHDR/" + hdrFramePrefix + (frame +
frameOffset).ToString("000");
    RenderSettings.skybox.SetTexture("_Tex", Resources.Load<Cubemap>(textureNameHDR));

    this.transform.position = new Vector3(float.Parse(cameraData[(int)frame][1]), float.
Parse(cameraData[(int)frame][2]), float.Parse(cameraData[(int)frame][3]));
110

    //if there is also rotation in the data file
    //If the incoming frames are pre-rotated, there is no camera rotation
    if (cameraData[(int)frame].Length > 4)
    {
115
        this.transform.rotation = new Quaternion(float.Parse(cameraData[(int)frame][4]),
float.Parse(cameraData[(int)frame][6]),
float.Parse(cameraData[(int)frame][5]),
float.Parse(cameraData[(int)frame][7]));
    }
120
}

void LoadCameraData()
{
125
    string allText = cameraMovement.text;
    string[] textLines = Regex.Split(allText, "\n");

    for (int i = 0; i < textLines.Length; i++)
    {
130
        string[] line = Regex.Split(textLines[i], " ");
        if(line[0] != string.Empty) cameraData.Add(line);
    }
}

void CreateReflectionProbes()
135
{
    if (camPathSegs <= cameraData.Count)
    {
        float segmentLength = (float)(cameraData.Count - 1) / camPathSegs; //no FOV in
camera data anymore
    }
}

```

```

140         for (int i = 0; i <= camPathSegs; i++)
        {
            int index = (int)Mathf.Round(i * segmentLength);
            reflectionProbesFrames.Add(int.Parse(cameraData[index][0]));
            reflectionProbesPositions.Add(new Vector3(float.Parse(cameraData[index][1]),
145         float.Parse(cameraData[index][2]), float.Parse(cameraData[index][3])));
        }

    }
    else //set reflection probe in every frame if more segments than number of frames were
    requested
    {
150         for (int i = 1; i < cameraData.Count; i++)
        {
            reflectionProbesFrames.Add(int.Parse(cameraData[i][0]));
            reflectionProbesPositions.Add(new Vector3(float.Parse(cameraData[i][1]), float
155         .Parse(cameraData[i][2]), float.Parse(cameraData[i][3])));
        }
    }

    GameObject[] reflectionProbes = new GameObject[reflectionProbesFrames.Count];
    for (int i = 0; i < reflectionProbes.Length; i++)
    {
160         int probeFrame = reflectionProbesFrames[i];
        reflectionProbes[i] = new GameObject("ReflectionProbeFrame" + probeFrame);
        ReflectionProbe probeComponent = reflectionProbes[i].AddComponent<ReflectionProbe
>() as ReflectionProbe;
        probeComponent.mode = UnityEngine.Rendering.ReflectionProbeMode.Custom;
        probeComponent.refreshMode = UnityEngine.Rendering.ReflectionProbeRefreshMode.
165         OnAwake; //update the probe only once, when it's created, for efficiency
        string textureName = "OmnidirectionalImagesHDR/" + hdrFramePrefix + (probeFrame +
        frameOffset).ToString("000");
        probeComponent.customBakedTexture = Resources.Load<Cubemap>(textureName);
        probeComponent.resolution = 256;
        probeComponent.size = new Vector3(20, 20, 20);
        reflectionProbes[i].transform.position = reflectionProbesPositions[i];
170     }
    }
}

```

B

APPENDICES FOR PETROL STATION RESEARCH STUDY

B.1 PARTICIPANT INFORMATION SHEET AND CONSENT FORM

Petrol Station User Study

*Required

Participant Information Sheet

This is the participant information sheet and consent form for participating in Joanna Tarko's (jkt26@bath.ac.uk), Checkmate VR (www.checkmatevr.com) and Dc-activ (dc-activ.com) study on understanding if online mixed-reality experiences produce a more realistic feeling and better recall than full computer-generated experiences.

Please read carefully before participating in the study.

Eligibility Requirements

You are eligible to participate in this study if the following applies to you:

1. You are at least 18 years old.
2. You are an active driver, where active means that you drove a car and filled it up with petrol in the past six months.

Description of the study

The study is a part of my doctoral project at the University of Bath, and is motivated by an interest in a degree of realism people associate with fully computer-generated environments vs mixed-reality environments.

If you agree to participate, you will be shown two short videos and you will complete an online survey, which will take approximately 15 minutes.

Participant's rights

You have the right to have your questions about the procedures answered. If you have any questions as a result of reading this information sheet, you should ask the researcher before the study begins.

You have the right to withdraw at any time without prejudice and without giving a reason.

Confidentiality/Anonymity

Your data will be collected anonymously. The data gathered from this study will be stored securely by Checkmate VR, Dc-activ and Joanna Tarko as the main research investigator of this project.

Benefits and risks

We do not anticipate that there are any risks associated with your participation. Your participation in this study is voluntary.

Upon completion of the study, if you provide us with your email address, you will be entered into a prize draw of two £50 Amazon vouchers. This is optional, and your email address will not be in any way associated with the collected data.

Data retention and publication

The data will be securely archived and retained for further research after this study finishes. Other researchers might be granted access to this preserved data for further research, providing that they agree to preserve confidentiality. Data extracted from the study may be used during presentation at conferences or published within academic papers.

For further information

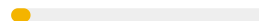
This study has been subject to the Department of Computer Science, University of Bath ethical review process. We will be glad to answer your questions about this study at any time. You may contact me at jkt26@bath.ac.uk

*

☐

I confirm that I have read and understood the information above, and agree to it.

NEXT



Page 1 of 12

Never submit passwords through Google Forms.

B.2 QUALIFYING QUESTIONS

Petrol Station User Study

***Required**

When was the last time you drove a car? *

☐ Today


☐ This week

☐ Less than a week ago

☐ Less than a month ago

☐ Less than six months ago

☐ More than six months ago

[BACK](#) [NEXT](#)  Page 2 of 12

Never submit passwords through Google Forms.

Petrol Station User Study

***Required**

When was the last time you filled up a car? *

☐ Today

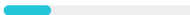
☐ This week

☐ Less than a week ago

☐ Less than a month ago

☐ Less than six months ago

☐ More than six months ago

[BACK](#) [NEXT](#)  Page 3 of 12

Never submit passwords through Google Forms.

B.3 DEMOGRAPHIC QUESTIONS

Petrol Station User Study

***Required**

What is your age?

Your answer

What is your gender?

☐ Female

☐ Male

☐ Prefer not to say

☐ Other:

How familiar are you with CGI graphics such as playing computer games? 1 means „I’ve never been exposed to any CGI graphics”, and 7 means „I’m very familiar with CGI graphics” *

1	2	3	4	5	6	7
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

The last time you purchased fuel what brand was the petrol station?

Your answer

The last time you purchased fuel did you pay at the pump or in store?

☐ At the pump

☐ In store

BACKNEXTPage 4 of 12

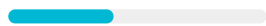
Never submit passwords through Google Forms.

B.4 FIRST VIDEO

Petrol Station User Study

First video

You will watch the first video now playing twice. It is recommended to watch it on Youtube in full screen mode. Imagine that you are driving your car to a petrol station to buy fuel. Press "Play" when you're ready. Do not replay the video.

[BACK](#)[NEXT](#)

Page 5 of 12

Never submit passwords through Google Forms.

B.5 PERCEPTIVITY QUESTIONS

Petrol Station User Study

*Required

What was the brand of the petrol station you visited in the video? *

*

Your answer

Can you remember the price of the fuel you wanted to buy? *

☐ Yes

☐ No

If yes, type the price below:

Your answer

How big was the petrol station you visited? *

☐ Small

☐ Medium

☐ Large

How many other cars are there? *

☐ 0

☐ 1

☐ 2

☐ 3

☐ 4

☐ 5+

Were you driving on the left or right-hand side of the road? *

- ☐ Left-hand
- ☐ Right-hand

How many buildings did you see excluding the forecourt and store? *

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5+

Which of these items do you remember in the journey? *

- ☐ Bus stop
- ☐ Traffic lights
- ☐ Blimp
- ☐ Airplane
- ☐ Petrol pump
- ☐ Streetlights
- ☐ Roundabout
- ☐ Pothole
- ☐ Fire extinguisher
- ☐ Poster stand
- ☐ Rubbish bin
- ☐ Give way sign
- ☐ Zebra crossing
- ☐ Speed limit sign
- ☐ Billboard
- ☐ Car wash
- ☐ Pelican crossing
- ☐ Lollipop man

BACK

NEXT

Page 6 of 12

B.6 ADDITIONAL PERCEPTIVITY QUESTION FOR MIXED-REALITY VIDEO

Did you notice any parts of the video that had been modified or inserted? If so, what? *

- ☐ Bus stop
- ☐ Traffic lights
- ☐ Blimp
- ☐ Airplane
- ☐ Petrol pump
- ☐ Streetlights
- ☐ Roundabout
- ☐ Pothole
- ☐ Fire extinguisher
- ☐ Poster stand
- ☐ Rubbish bin
- ☐ Give way sign
- ☐ Zebra crossing
- ☐ Speed limit sign
- ☐ Billboard
- ☐ Car wash
- ☐ Pelican crossing
- ☐ Lollipop man
- ☐ Other: _____

[BACK](#)[NEXT](#)


Never submit passwords through Google Forms.

B.7 FIRST VIDEO PLAYED ONCE MORE

Petrol Station User Study

First video

Now watch the video again. It is recommended to watch it on Youtube in full screen mode. Press "Play" when you're ready. Do not replay the video.



BACK

NEXT

Page 7 of 12

Never submit passwords through Google Forms.

B.8 PRESENCE QUESTIONNAIRE

Petrol Station User Study

*Required

Please rate your sense of being in the car driving to a petrol station, on a scale of 1 to 7, where 7 represents your normal experience of being in a place.

I had a sense of “being there” in the car driving to a petrol station: *

	1	2	3	4	5	6	7	
Not at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very much

To what extent were there times during the experience when the video was the reality for you?

There were times during the experience when the petrol station was the reality for me... *

	1	2	3	4	5	6	7	
At no time	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Almost all the time

When you think back to the experience, do you think of the video more as images that you saw or more as somewhere that you visited?

The petrol station seems to me to be more like... *

	1	2	3	4	5	6	7	
Images that I saw	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Somewhere that I visited

During the time of the experience, which was the strongest on the whole, your sense of being in the car driving to a petrol station or of being elsewhere?

I had a stronger sense of... *

	1	2	3	4	5	6	7	
Being elsewhere	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Being in the car

Consider your memory of watching the video. How similar in terms of the structure of the memory is this to the structure of the memory of other places you have been today? By 'structure of the memory' consider things like the extent to which you have a visual memory of the virtual environment, whether that memory is in colour, the extent to which the memory seems vivid or realistic, its size, location in your imagination, the extent to which it is panoramic in your imagination, and other such structural elements.

I think of the petrol station as a place in a way similar to other places that I've been today... *

	1	2	3	4	5	6	7	
Not at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very much so

During the time of your experience, did you often think to yourself that you were actually in the car driving to a petrol station?

During the experience I often thought that I was really driving a car... *

	1	2	3	4	5	6	7	
Not at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very much so

[BACK](#)[NEXT](#) Page 8 of 12

B.9 GENERAL PREFERENCE QUESTIONS

Petrol Station User Study

*Required

Which video did you like better? Why? *

Your answer

In which of the two environments would you be more likely to behave like at a normal petrol station? *

- ☐ First
- ☐ Second

Did you notice anything weird about the videos? If yes, what it was? *

Your answer

BACK

SUBMIT

Page 12 of 12

Never submit passwords through Google Forms.