Lukas Fritz, Bsc

HDR Light Probes for Mixed Reality Environments

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. David Mandl
Ass. Prof. Dipl.-Ing. Dr.techn. Denis Kalkofen

Institute of Computer Graphics and Vision
Head: Univ. Prof. Dipl.-Ing. Dr.techn. Dieter Schmalstieg

Graz, October 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

01.10.2020	Lukas Fritz
Date	Signature

Acknowledgements

I would like to express my thanks to my supervisor Ass. Prof. Dipl.-Ing. Dr.techn. Denis Kalkofen for the helpful comments, and remarks during the conduction of this master thesis.

I would particularly like to thank Dipl.-Ing. David Mandl for his regular meetings, during those he suggested numerous improvements to both the content and presentation of this thesis and whose own image-based lighting project has contributed to these results.

Special thanks to Karl Voit for providing this LaTeX template.

In addition, I would like to thank my family, friends, and many others for their valuable input and support, including David, Gabriel, Philipp, Pia Karoline, Saara, and Thomas.

Kurzfassung

Image-Based Lightning (IBL) Methoden in Kombination mit High-Dynamic-Range (HDR) Bildern ermöglichen eindrucksvolle Erfahrungen in Mixed Reality (MR) Szenarien. Realistisches Shading von virtuellen Modellen mit einer lebensechten Beleuchtung führt zu einer glaubwürdigen Fusionierung von realen und gerenderten Objekten.

Die Aufnahme von HDR Video Information ist üblicherweise nur mit teuren Geräten und Ausstattung möglich, worauf die meisten Nutzer keinen Zugriff haben. Hier wird eine neuartige Lösung präsentiert die eine Kombination von Algorithmen aus verschiedenen Computer Vision Bereichen verwendet: Feature Matching, Image Warping, Optical Flow, und Graph-Cuts, welche vergleichbare Resultate erzeugen.

Der Hauptunterschied zu anderen Forschungsprojekten ist die Verbesserung eines Standard-Dynamic-Range (SDR) Zielbildes mit HDR Information und die Anwendung dieser erzeugten HDR Information in einer IBL-basierenden MR Applikation.

Unsere Methoden werden qualitativ und quantitativ verglichen mit idealen Resultaten sowie mit der SDR Beleuchtung, in drei verschiedenen Beleuchtungsszenarien und unterschiedlichen 3D Modellen.

Da die erzeugte HDR Information nur als Lichtquelle angewandt und nie direkt angezeigt wird, führen sogar nicht ideale Resultate zu schlüssigen Ergebnissen die eine Verbesserung zu üblichen Methoden darstellt, und besonders bei diffusen Objekten herausragt. Unsere Lösung ist als leistbare Alternative für CGI Film Beleuchtung sowie im MR Bereich anwendbar.

Abstract

Image-based lighting (IBL) methods paired with High-dynamic-range (HDR) images enable an immersive experience for mixed reality (MR) scenarios. Realistic shading of virtual models with real-world illumination may lead to a credible fusion of reality and rendered objects.

Capturing HDR video information is usually linked to expensive equipment and hardware, which ordinary users may not have access to. Here, a novel solution that offers comparable results is presented utilizing a combination of algorithms from different computer vision fields: feature matching, image warping, optical flow, and graph-cuts.

The main difference to other research is the enhancement of Standard-dynamic-range (SDR) target frames with HDR information and applying the created HDR information in an MR application using IBL.

Our methods are qualitatively and quantitatively compared to ideal results and SDR illumination, featuring three different lighting scenarios and various 3D models.

As generated HDR information is only applied as a light source and not directly displayed, even non-ideal results lead to coherent results improving conventional methods, excelling on diffuse objects. Our design is therefore applicable as an affordable alternative for CGI movie illumination and MR.

Contents

List of Figures

1 Introduction

In Augmented Reality (AR) and Virtual Reality (VR) applications, the user is either put into an enhanced version of the real world or a completely artificial place in a virtual environment. In both options, the user is presented an addition of virtual content. These applications can either be used for professional content or simply for entertainment value. Popular examples are *Pokemon GO* [64], *Half-Life: Alyx* [104] or applications like Snapchat [43] or Instagram [26] which layer various AR filters on top of the camera, e.g. masks, frames or artwork.

(a) VR application: Half-Life: Alyx [104]

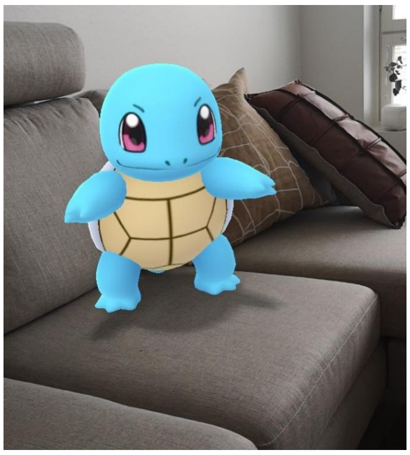

(b) AR application: Pokemon GO [64]

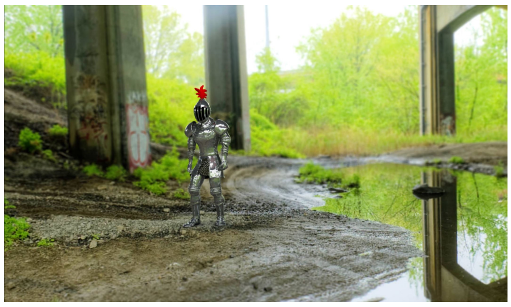
Figure 1.1: Example Virtual- and Augmented-Reality applications

To provide the user with an immersive experience, multiple factors influence the perceived quality, like illumination, size, or physical properties, with the ultimate goal that the user should not be able to distinguish the virtually created content from the real world [62]. While all these quality factors are of high importance, this thesis focuses on the area of virtual lighting.

To create a result that is as realistic as possible, it is first necessary to gather information from the scene which should be enhanced in the final version. In this work, it was decided to make use of 360° High Dynamic Range (HDR) images for reproducing realistic lighting conditions of the real scene. HDR displays the deepest shadows and the brightest highlights of a scene, as each image consists of several pictures at a different exposure time. Therefore, it excels at depicting illumination cues and serves ultimately as input for a system that applies image-based lighting (IBL) onto a virtual object in an AR-application.

Present research often provides solutions for professional settings, requiring special hardware, like HDR cameras or extremely strong computers, which is most often impossible to afford as a private consumer. Thus, a solution is presented by using a combination of images and algorithms to enable users with accessible hardware to reach similar results. Although it may seem that everyday users have no use for this modern technology, several devices already support HDR content, or are at least experimenting with this technology. For instance, the Samsung Galaxy S10 referred to as HDR10+ [37] or the BenQ EX3501R monitor [18].

Also the (indie) film industry is a potential target area, as photo-realistic illumination paired with computer-generated imagery (CGI) can lead to excellent results, while still being affordable.


(a) CGI with IBL by a private media artist [34]

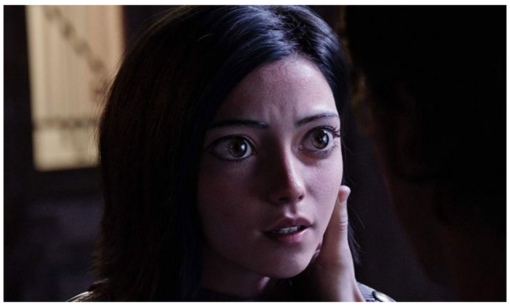

(b) CGI in the movie Alita: Battle Angel [29]

Figure 1.2: CGI in an indie project versus professional film industry

1.1 Goals

This thesis presents a novel solution creating a realistic mixed reality (MR) experience using 'normal' Standard Dynamic Range (SDR) recordings of a scene paired with the information of sparse HDR images.
It aims to remove the constraints of requiring expensive hardware, while still creating reasonable results, especially in comparison with non-IBL solutions. The main goal is to demonstrate how a combination of algorithms and affordable hardware performs concerning lighting compared to a high-end solution utilized by professional film studios. Thereby, the main hypothesis is whether the conduction of qualitative and quantitative analysis leads to the generation of promising results in the majority of recorded scenarios. For this purpose, expensive hardware is compensated with the combination of feature matching, optical flow, tone-mapping, and graph-cut algorithms, while still reaching results of similar quality.

Another focus of this project lies in the utilization of SDR frames while feature matching, and applying this matching information in the warping part in combination with HDR panoramic-frames. Subsequently, these created light probes can then be applied as a source of lighting information to illuminate for example a virtual object, as HDR excels in this case and serves realistic outcomes. IBL utilizes the accurately calculated real-world information to recreate the scenes lighting rather than modeling the illumination.

Realism is a major goal for any AR application, ultimately achieving that real and virtual objects are indistinguishable. To do so the user has to create an AR, VR, or other MR application and utilize the HDR information as a lighting source for e.g. diffuse or specular shaders.

The final goal of this thesis is the creation of the mentioned mobile MR application featuring dynamic IBL by utilizing the created HDR images and comparing its performance to existing solutions.

1.2 Challenges

As HDR is still a quite new technology and only a handful of the current systems support HDR at the moment, difficulties have been arising while using technology which is not yet considered as standard. For example, in many required algorithms HDR was not yet supported. These circumstances were the reason for creating a workaround by using SDR images for matching and applying those values. During the research on related work, no existing solution which features this kind of workaround concerning HDR warping was determined.

To begin with, one of the obstacles has been the lack of an HDR display or monitor. As only some of the latest monitors can display HDR images, many users would have to tone-map the given or resulting HDR-frames to interpret them correctly. This is not only inconvenient but additionally slows down the evaluation process, as a user cannot tell with the naked eye if the result differs a lot from the actual frame. HDR images show a bigger range of brightness values compared to an SDR image which is either too light or too dark, in all cases lacking details.

Surprisingly neither on Windows 10 nor Mac OS, the pre-installed image viewers were able to display output from any HDR-frames, except a white screen. The Luminance HDR tool [56] and Picturenaut [36] enabled the creation or simply viewing of the HDR-images, with the option to manually adjust the brightness of an image and even tone-map them as a preview. On the provided Samsung Galaxy S9, the phone claimed to be unable to open the '.HDR' filetype and the Google Playstore did not provide any HDR viewers, just mainly creation tools. With the Android application called 'MPlayer' HDR videos could be played back with the setting 'Software encoding'.

An additional difficulty is the lack of HDR-technology in modern software, forcing the adaption of source code by researchers and engineers to gain access to modern technologies, as they only sparsely exist in standard software like Matlab. This environment provides an HDR read-and-write-function, but only handles the '.HDR' file extension and none of the other file formats like '.EXR'. Further functions exceeding read-and-write are absent in the current Matlab distribution, which is expected to change in the near future.

Due to the fact that the provided 360° camera (see section 3.1) did not support an exposure series, the images have to be recorded on each location where an HDR image should be captured. Depending on the size of the scene, the amount of required images also scales, as the whole scene should be covered as HDR data (for details see subsection 3.1.1).

A significant limitation of mobile devices is the low amount of available storage and graphic memory. Moreover, the processing power is much less than a modern desktop computer with an extra graphic processing unit (GPU). As mobile phones still struggle with the data amount of high-quality HDR images, the resolution of the images had to be reduced, and the HDR frames could only be applied for IBL.

2 Related work

This chapter gives an overview of already existing research that is related to the presented thesis (i.e. HDR, IBL, feature matching, optical flow, and graph-cut) and additionally how featured technologies are used in other research projects.

The main difference to other research is the enhancement of an SDR target frame with HDR information and applying the created HDR information in an MR application featuring IBL.

2.1 Definitions and keywords

For each of the system's parts several algorithms are involved to achieve the final result. Therefore, the following sub-chapters explain the related techniques, computer graphics, and -vision concepts and terms connected to this thesis. Additionally, fields of application of each of the concepts are presented.

2.1.1 High dynamic range (HDR)

As HDR is generally regarded as providing enhanced quality (vs. SDR images), the next chapter is giving up-to-date information on its characteristics. In brief, HDR images are characterized by a greater bit depth, a wider range of luminance, and a more extended color volume in comparison to SDR images.

HDR image formats There are several formats for storing HDR images, differing in the usage of memory space. The non-zero part of a number is called the *Mantissa*, whereas the Exponent defines how many positions the decimal point is moved. Additionally, there can be a *sign bit* to show if a value is positive or negative. These elements are defining the floating-point number, utilized to encode the high precision of HDR values.

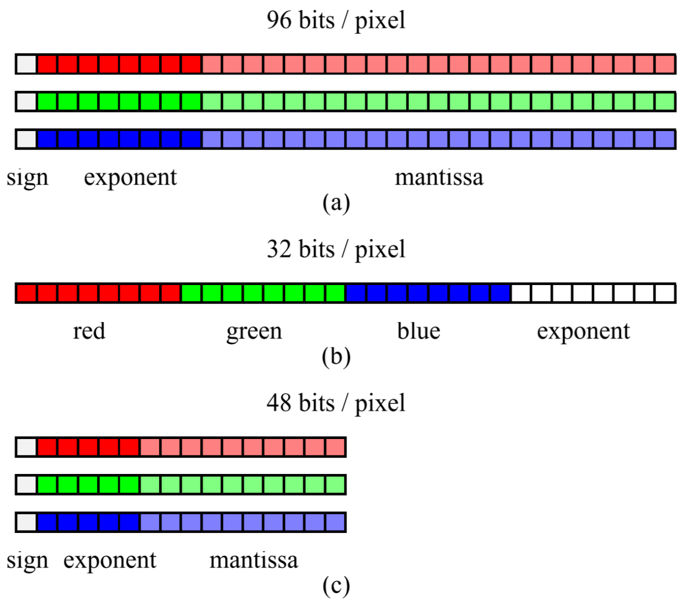


Figure 2.1: HDR encoding types: (a) Portable PixMap (.ppm) (b) Radiance (.pic, .hdr) (c) OpenEXR (.exr) [85]

Portable PixMap (.ppm) for example uses for all the red, green, and blue values a 32-bit IEEE float (see 2.1.a), which is the recommended option if storage space is not a limiting factor. Alternatively, there is the more compact Radiance format (.pic, .hdr) applying a single common exponent and a mantissa for each channel(see 2.1.b). Lastly, OpenEXR (.exr) is supported natively by most modern GPUs and uses 16 bit floats for each color channel (see 2.1.c).

Acquisition of equirectangular HDR Images As the provided 360° camera does not support the native recording of HDR images, the method known as exposure bracketing was used, which is shown in Figure 2.2.

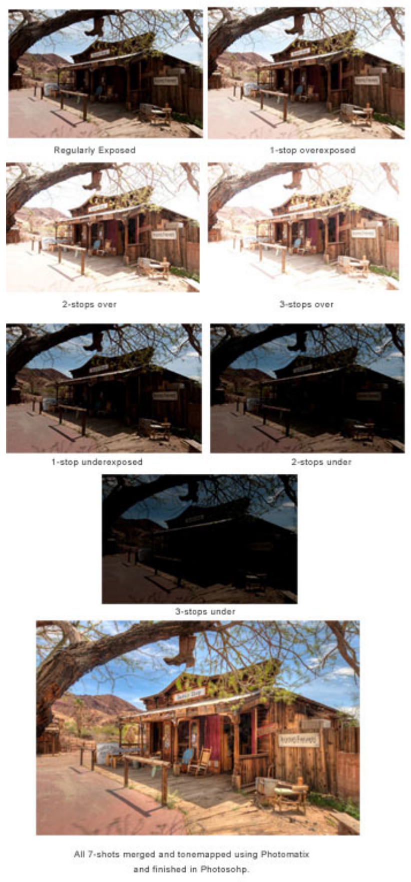


Figure 2.2: Example exposure series to create an HDR Image [99]

In each position where HDR Images should be recorded the user has to take several images at the same set of exposure times. Moreover, the white balance of the camera should be locked. It is important to set the camera to *Manual mode*: in any other mode, the automatic exposure function would adjust the exposure between shots. Thereby, parts of the scene with light sources in it would be taken at a different set of exposure time from the darker parts of the panorama [100].

A tripod is suggested, but not required. At least a stable surface is highly recommended to reduce artifacts. It is also vital that the scene stays identically, which is a challenging task in dynamic scenes as every motion leads to artifacts, like ghosting. Also changing illumination can create problems during the creation part, for which the Picturenaut software (section 4.5) was utilized.

2.1.2 Tonemapping

The tone mapping algorithm is mandatory if the light intensity level or contrast experienced in a real environment exceeds the available dynamic range of the display device. However, simple downscaling of the illumination- or contrast range does not produce satisfying or accurate results. During the past decades color and image appearance models have been created to reproduce the visual appearance which the human eye experiences in the 'real world'.

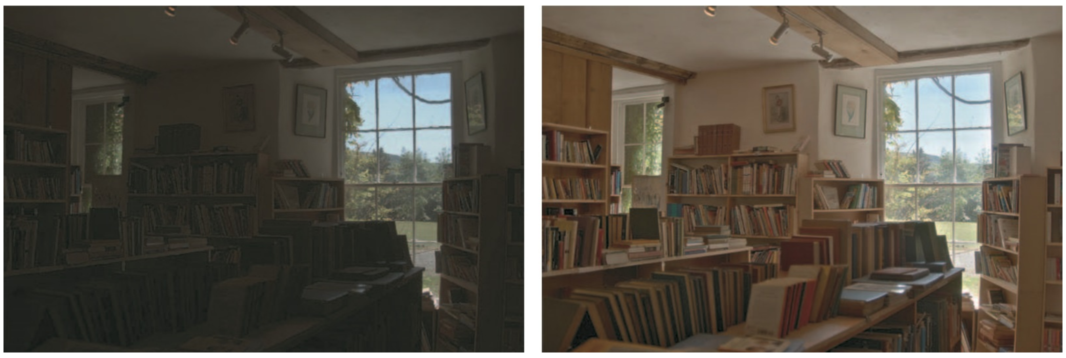


Figure 2.3: HDR image: Linear scaling versus tone-mapped result [75]

Applications Tone mapping has several application fields like producing optically pleasing images, reproducing as many details as possible, maximizing contrast, or creating visual effects for image editors. In scientific projects, it is mainly used to adjust the dynamic range to match the available one, as those might not always be fitting together.

During this project tone mapping was applied to generate SDR frames by tone mapping HDR frames, display them on any device, print them within this thesis and especially make them usable in the necessary algorithms and tone map the resulting images.

2.1.3 Augmented Reality

AR is defined as '*a technology that layers computer-generated enhancements on top of the existing real world to make it more meaningful through the ability to interact with it*' [1]. Simplified, AR enhances the real world with virtual models and information. Many current applications are based on marker-based technology, as it is much simpler to detect and recognize images that have been hard-coded in a device in advance.
Contrary to that a markerless AR application can recognize almost any object, which needs a higher amount of effort as it is more difficult concerning the implementation [1].

Differences to Virtual Reality Both AR and VR are relatively similar technologies with mainly the same purpose, namely to serve users with an enhanced or enriched experience. The main property of VR is the creation of a virtual world that is completely computer generated and driven. On the other side, AR adds virtual components as a new level of interaction to the real world. Therefore the only requirement for AR is that the device includes a camera, whereas in VR the user needs an own headset for displaying the whole virtual scene [80].

Mixed Reality MR does not exclusively take place in either the physical or virtual world but is a mixture of both [60].
MR is best explained as '*the range between reality and VR, which allows virtual and real elements to be combined in different degrees [80]*'.

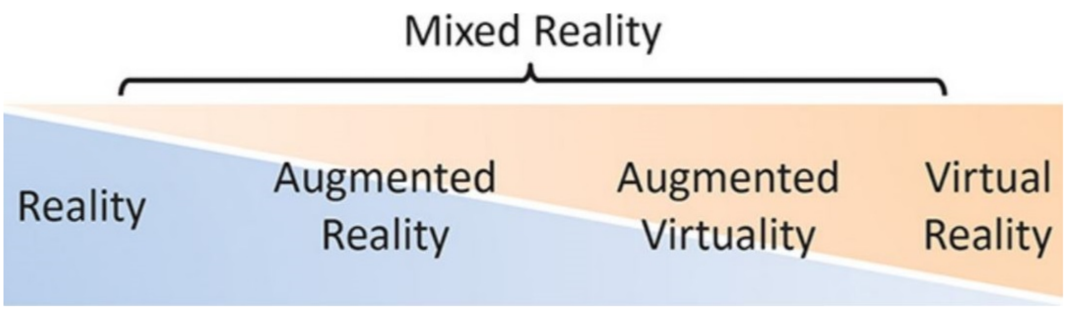


Figure 2.4: The so called 'mixed reality continuum' contains all possible combinations of the real and virtual worlds [60].

Applications Applications for AR are for example available in the entertainment sector, like mobile games or animations (see Figure 1.1), in the medical field, as guidance or overlay for areas where the human eye lacks information, in the military, as an overlay of additional information for troops, in the navigation field to localize certain buildings or add information to sights and in the educational sector to teach with augmented models or scenes [14]. In this thesis, AR is utilized to render a virtual object in a real scene with fitting HDR illumination of this virtual model.

2.1.4 Global Illumination - Image-Based Lighting (IBL)

In real life, the light originates from every direction either directly from light sources or indirectly after reflecting off objects in the environment, being partially absorbed in the process. The whole real scene can be seen as a single light source for virtual objects, which leads to more realistic and lifelike illumination as it adds a higher amount of details to the objects (see section 5.1). HDR-images, in particular cube-maps, are a great way to encode such an 'environment lighting'. This technique is referred to as *Image-Based Lighting* or sometimes *Indirect Lighting* [31].

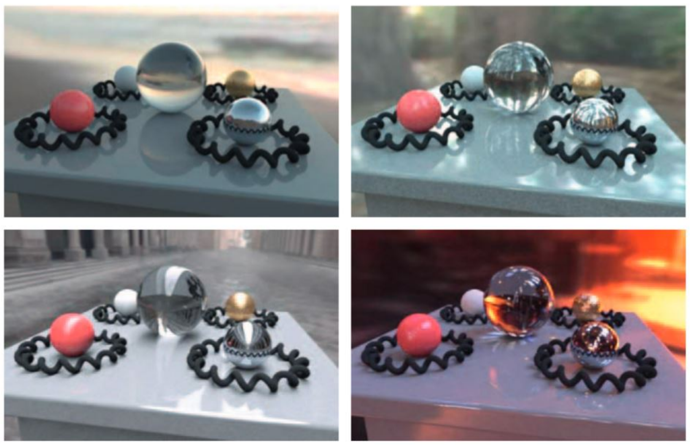


Figure 2.5: HDR images used to light an artificial scene [74]

It is usually mandatory to utilize HDR for IBL as the illumination in environment maps has to be in floating-point precision, rather than arbitrary fixed-precision values to ensure meaningful physical units. Connected to these maps two distinct types of environment mappings exist. On the one hand, if the map represents incoming light from the point of view of an observer it is called *radiance map*, and on the other hand, the type of map that directly represents outgoing light after reflection which is referred to as *irradiance map* [80].

Reflectance and environment lighting The basic illumination model in computer graphics consists of three components: Ambient light, diffuse reflection and specular reflection. Ambient light models the combination of light reflections, which origin from surrounding objects in the scene. It has neither spatial nor directional characteristics and can be generally seen as background lighting. Diffuse reflection is designed for matte surfaces. The resulting surfaces appear equally bright from any viewing angle as they are modeled to reflect light with the same intensity in every direction. Specular reflections occur on any shiny surface and their appearance depend on the objects material and the light. Contrary to diffuse reflection they model reflected light towards the camera [40].

To create an environment map the integral for each output ω_o sample direction has to be solved by sampling a high amount of directions ω_i over the hemisphere Ω while their radiance is averaged. The sample directions ω_i from the hemisphere are oriented towards the output ω_o sample direction [92].

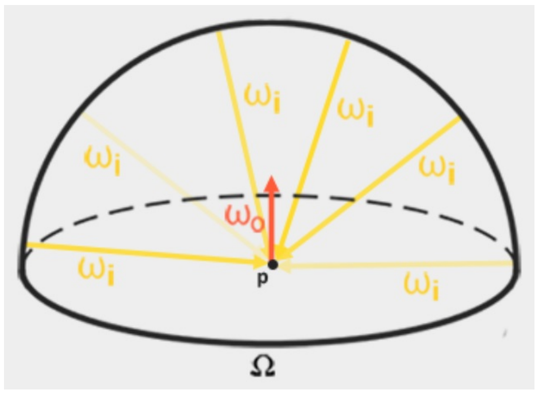


Figure 2.6: Simple sketch for environment lighting [92]

In Equation 2.1 the reflectance equation is presented with the main goal of '*solving the integral of all incoming light directions ω_i over the hemisphere Ω*' [17].

$$\begin{gathered}L_o(p,\omega_o) = \int_\Omega (k_d \frac{c}{\pi}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i + \\ \int_\Omega (k_s \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}) L_i(p,\omega_i) n \cdot \omega_i d\omega_i\end{gathered} \tag{2.1}$$

The diffuse k_d and the specular k_s term are independent of each other therefore the integral is presented split into two parts, to handle them individually [17].

The BRDF equation is used to define how light is reflected on the surfaces of given materials. Many reflection models use their own BRDFs, which vary in their properties like complexity and level of realism, e.g. Phong [72], Oren–Nayar model [67], Cook–Torrance model [17], and others. Phong shading, which is seen as a classical model does not provide a very realistic illumination. It still creates appropriate outcomes, but lack in realism. In contrary, physically-based rendering (PBR) approaches as seen in the Oren–Nayar model and the Cook-Torrance model aim for authentic representations of materials. The Oren–Nayar reflectivity model is designed for diffuse reflection from rough surfaces whereas the Cook-Torrance model specializes in specular PBR.

In realistic natural lighting scenarios, as part of this thesis, the color channels' brightness values are typically above 255 and contain more accurate values than fixed-point numbers. Therefore physically-based BRDFs should always be paired with HDR to reach optimum results.

The software *Unity* (see section 4.4) with which the AR application was created, utilizes the Torrance–Sparrow model, which is a PBR model representing the directional reflectance characteristics of a specularly reflecting rough surface [87].

Generally, IBL is independent of the type of BRDF but creates more lifelike results if it is paired with a physical-based BRDF, as the light sources are represented as images from the real world. Usually pre-processing concerning reflection and lighting is applied in connection with IBL, to improve performance during execution. This procedure differs concerning whether the light probe is used for diffuse or specular reflections. For diffuse reflection, every point in the hemisphere gets blurred in advance to generate a low-frequency illumination, which averages the color in the areas. Whereas for specular reflection, a pre-computed environment convolution map with the focus on roughness is created and the sequentially blurrier results are stored. Additionally, the BRDF's response to various light directions featuring several different roughnesses of the material is pre-computed [92].

Light sources and alternatives In computer graphics there are typically three variants of light sources available: point lights, directional lights, and spotlights.

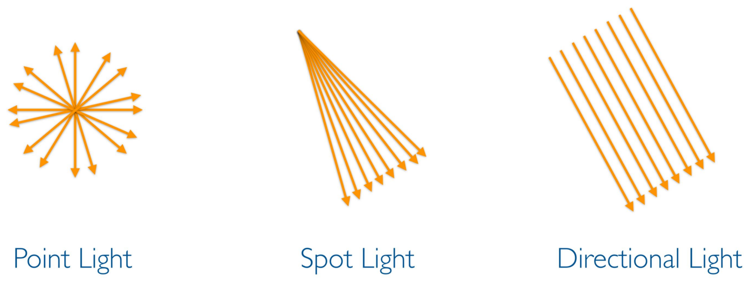


Figure 2.7: Schematic of light source types in computer graphics [106]

A point light distributes light equally in all directions, simulating a naked bulb suspended from a ceiling without any shade. A spotlight is similar, but it simulates a theatre spotlight, which spreads light uniformly but restricts it to a cone, which is gradually less intense towards the sides of the cone. Otherwise, a directional light imitates an infinitely distant light source, illuminating the objects in the scene from a single direction, with equal intensity [40].

IBL and the other light sources are not exclusive to each other. Often a combination of several of them leads to the most realistic representation of scene lighting. In computer games for example IBL is often utilized for environmental lighting, whereas a spotlight is chosen to simulate a flashlight and point sources to mimic the behavior of light bulbs. It is not always necessary to apply IBL, as specific scenes like dark caves contain hardly any light sources except e.g. a flashlight or a torch.

2.1.5 Feature extraction and matching

Feature extraction and matching is a highly relevant part of computer vision as it is needed in various commonly known algorithms and tasks, such as object detection, robot navigation, Structure-from-Motion (SFM), and other image related tasks. An image consists usually of a large variety of features, although not all of them are regarded as useful. Favorable feature points are characterized by four properties: 1) repeatability, 2) saliency, 3) locality and, 4) compactness and efficiency [35].

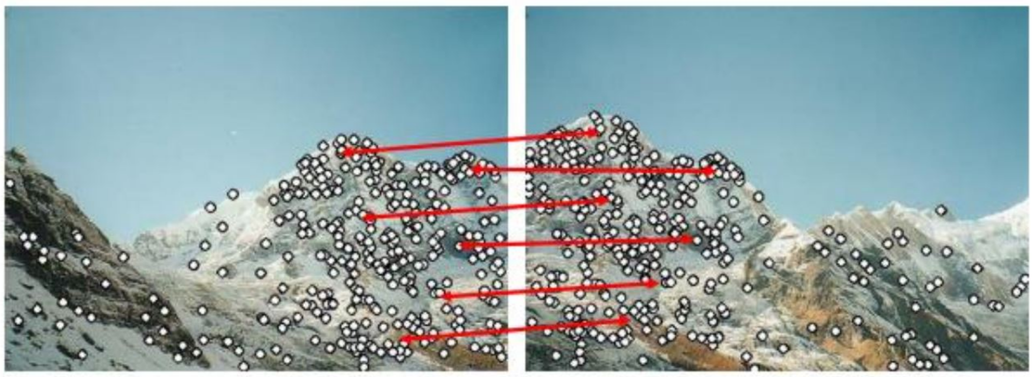


Figure 2.8: Example feature- extraction and matching [35]

Typically, the feature extraction and matching process comprises of four phases [89][90]:

- feature point detection - identification of the interest points
- computing feature descriptors for each of them - creation of a description vector for each feature point. Each feature point descriptor is rotation, scale, translation and lighting invariant.
- matching of corresponding keypoints - feature point descriptors are checked between the images, to identify matching keypoints. Typically they are linked as keypoint pairs (Ai, Bi) - (Ai', Bi'), where (Ai, Bi) is a feature point in the first image and (Ai', Bi') is the fitting match in the second image.
- transformation estimation between the two pictures utilizing the matching keypoints

Those four steps are named in order of the process, therefore the detection step is the most consequential of those points, as it affects all further tasks.

Applications There are many different regions of application in modern technology for feature detection and matching, for example:

- Automate object tracking
- Image alignment

- Stereo calibration
- Image segmentation
- Image classification
- Robotic mapping and navigation
- Image indexing and content retrieval

During this project, feature detection was applied to both tone-mapped HDR images and the target SDR frame, and following that, feature matching was then applied to identify corresponding points in both of them. This was necessary as both images were shot from a different location, angle, and rotation. Later those matched features were used to warp the HDR image into the perspective of the SDR frame.

2.1.6 Optical Flow

Szeliski [85] described optical flow as: '*The most general and challenging version of motion estimation is to compute an independent estimate of motion at each pixel. This generally involves minimizing the brightness or color difference between corresponding pixels summed over the image*'.

In the following figure, a simple sketch of the optical flow of three consecutive frames is displayed.

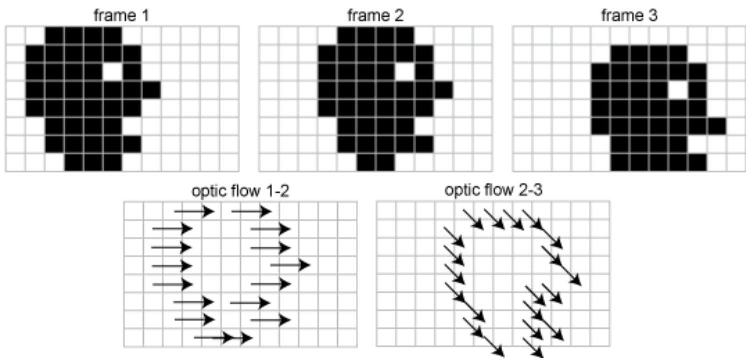


Figure 2.9: Detection of optical flow on the image plane [22]

Applications Correct estimation of optical flow may be utilized for motion-related detection and tracking. It is a core tool of computer vision for the quantification of perceived changes or motion using image sequences [42].

In this project, the optical flow algorithm by C. Liu [52] was applied to calculate a new HDR light probe between two adjacent HDR images: one forward warped and one backward warped image. These newly created images are forwarded as input to the graph cut algorithm (see subsection 2.1.7).

2.1.7 Graph-cut

As the optical flow function of the framework calculates the previously mentioned backward and forward flows it provides two output frames. Those two images are needed to combine to a single frame which should represent the optimal mixture of all of them, in terms of appearance. That is the point where the graph-cut comes into play, as its task is the combination of the two mentioned images, with additional information of the SDR image.

During this project, graph-cut has been connecting patches from either the left warped HDR frame or the right warped HDR frame, as the SDR images would create too distinct results. To avoid the influence of the SDR frame during the graph-cut the SDR frame has been penalized with a high weight of 500, to assure rather a patch from any adjacent HDR frame is used. This weight is the variable c in Equation 2.2.

To calculate the data cost of graph-cut, the following equation has to be fulfilled to decide which patch is chosen for the following blending. As part of this equation, the calculated motion confidence is required, is calculated during the optical flow for each pixel [33].

Equation 2.2 shows the formal definition of the data cost function D for computing the cost of assigning a given label l to a pixel p [33].

$$
\begin{aligned}
D(p,l) &= \begin{cases} c & \text{if } l = f_i \\ \infty & \text{if } w_l(p) \\ D_c(p,l) + D_f(p,l) + D_d(l,f_i) & \text{undefined} \quad \text{otherwise} \end{cases} \\
D_c(p,l) &= \|w_l(p) - f_i(p)\| \\
D_f(p,l) &= 1 - \text{motion\_confidence}(w_l(p)) \\
D_d(l,f_i) &= \frac{|\ \text{frame\_index}(w_l) - i\ |}{|\ \text{frame\_index}(w_{right}) - \text{frame\_index}(w_{left})\ |}
\end{aligned}
\tag{2.2}
$$

- The constant c is the defined as cost for choosing the SDR option as target pixel, which in this case was set to 500 to avoid any SDR information in the resulting picture;
- w_l is the warped image (either left or right);
- D_c is applied to improve color consistency between a pixel and the label;
- D_f is applied as factor to the motion vector confidence;
- D_d favors labels that are temporally closer to the target frame i.

The pixel channel- and the confidence-values are both numbers between [0-1]. The motion vector confidence is generated during the optical flow computation.

Applications Applications for graph-cuts are for example image and video texture synthesis like in the project of Kwatra *et al.* [48], which transformed and copied region from one scene to another, or generated additional textures without any perceptual inconsistencies.

2.2 Related papers

2.2.1 HDR-Imaging

In 1997 Debevec *et al.* [20] described a process of retrieving HDR radiance maps from ordinary photographs recorded by standard cameras. As seen in Figure 2.10, several conventional images are recorded of the identical scene varying only the value of exposure. Following that, the image series may be algorithmically combined to an HDR image.

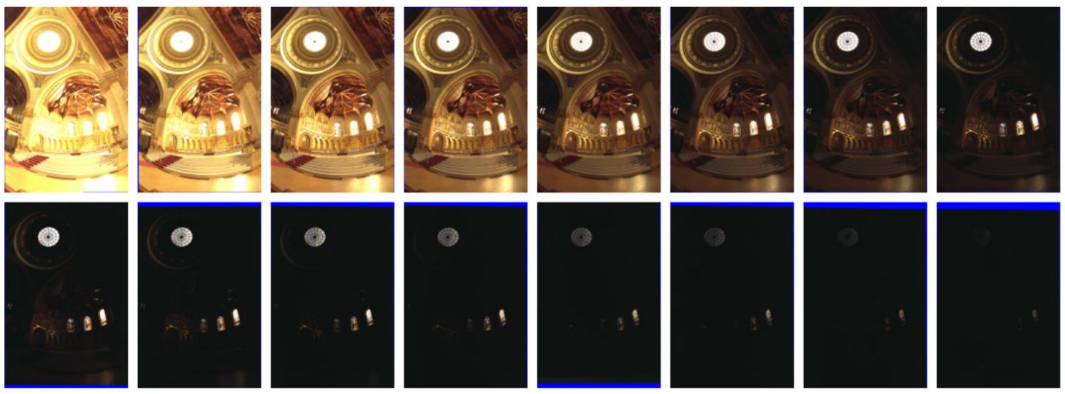
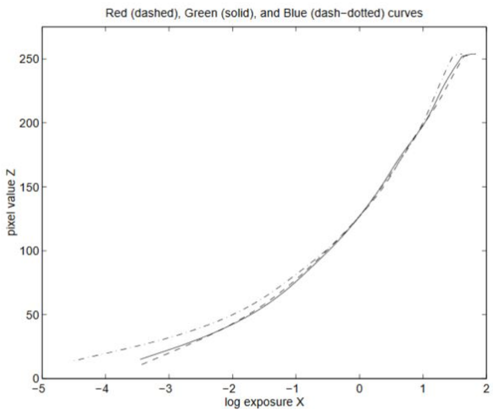
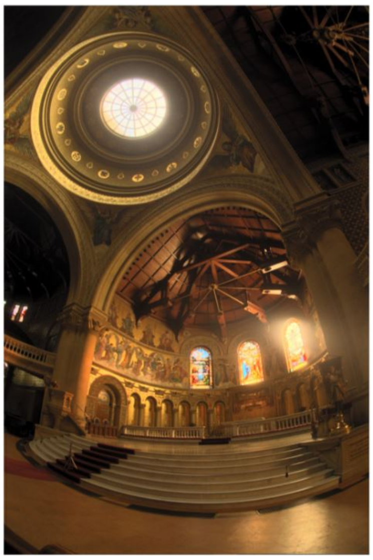


Figure 2.10: Sixteen photographs of a church with various exposure [20]

The algorithm recovers the response function (Figure 2.11) from the multiple photographs and utilizing this function fuses them into a single HDR radiance map. The pixel values in this resulting map correspond to their correct values (Figure 2.11). This method of HDR Image creation is still used nowadays and it was moreover applied in the practical task for this thesis. For further details see subsection 2.1.1. For this project, the Reinhard05 operator [75] was chosen as tone mapping-operator to convert HDR images into the SDR domain. Although it is regarded as one of the more simple operators, it is at the same time one of the fastest and most widespread ones, leading to fine results. There are specialized research papers, like '*A comparative review of tone-mapping algorithms for high dynamic range video*' by Eilertsen *et al.* [24] comparing all pros and cons of each of these tone mapping-operators, as there is a large amount of them available.



(a) Recovered response function for the imaging system used in the church photographs in Figure 2.10 [20]



(b) Resulting rendering using histogram compression also simulating the human visual system, e.g. glare [20]

Figure 2.11: Utilized response function and the resulting reconstructed HDR radiance map

Concerning the acquisition of HDR images one publication in 2019 by Yan *et al.* [101] deals with progress in the 'ghosting free' creation of them through an '*attention-guided end-to-end deep neural network (AHDRNet)*'. This project shows a non-flow based solution which avoids errors and artifacts due to optical-flow estimation.

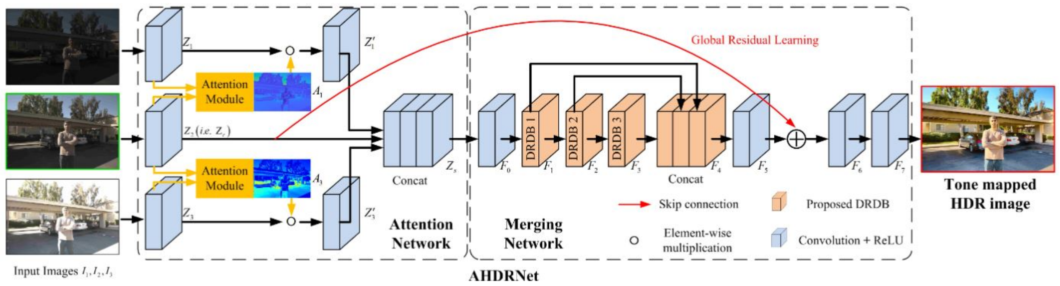


Figure 2.12: The AHDRNet system consists of two networks. An attention network to analyze and extract features plus a merging network for HDR estimation[101]

Other recent researches show trends in reconstructing HDR images from a single SDR image by utilizing deep learning. The approach by Lee *et al.* [49] aims to create several exposures of a single input SDR image by utilizing a multi-layered convolutional neural network (CNN) (see Figure 2.13). Following that, those images may be fused as typically, to create the target HDR image. Similarly, Khan *et al.* [44] apply the idea of feedback in their recurrent neural network with a hidden state.This enables them to reconstruct a target HDR frame by guiding their low-level features of the image utilizing extracted high-level features in the mentioned hidden state of the system.

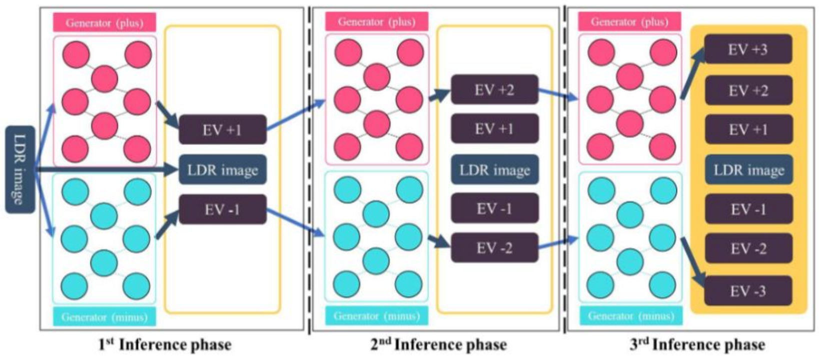


Figure 2.13: CNN based exposure-bracket generation to generate an HDR utilizing only one SDR target image [49]

As HDR images tend to have much larger file sizes, compression of HDR content is another interesting topic. In the recent paper from 2019, Maymon *et al.* [57] applied contrast optimization and local adaptation connected to HDR and could drastically compress the dynamic range, without eliminating the existing fine details of the HDR content.

2.2.2 Image-Based Lighting

In 1998 Debevec [21] introduced image-based lighting with HDR and differential rendering, which is the method that allows real-world illumination effects to be preserved. Although initially in 1984 Miller *et al.* [61] discussed environment maps that can be applied to perform image-based diffuse and specular lighting. Paul Debevec greatly influenced the related work with his influential research throughout the years, and his name occurred in many scientific works connected to this thesis.

An extremely interesting approach by Meerbeek *et al.*[7] describes a method of searching an image database for images to be used in rendering a scene using IBL (Figure 2.14). Thereby, the user searches for images including lighting infrastructure information additionally to keywords. The results are based more on the lighting infrastructure rather than the user's word-based search terms.

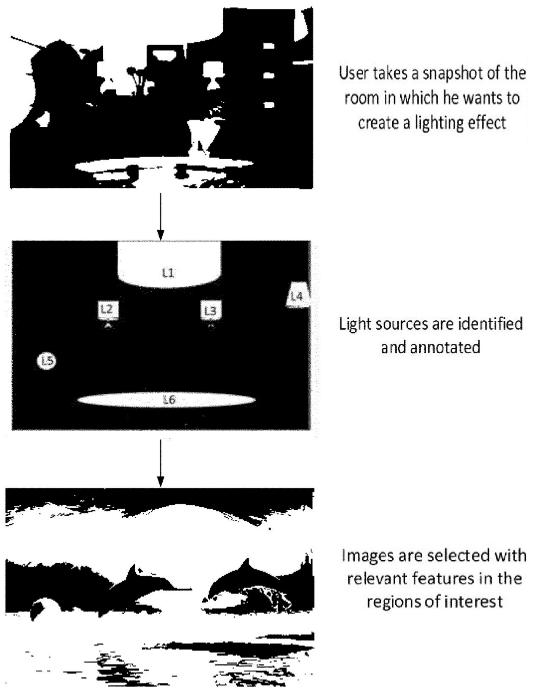

User takes a snapshot of the room in which he wants to create a lighting effect
L1
L2
L3
L4
L5
L6
Light sources are identified and annotated
Images are selected with relevant features in the regions of interest

Figure 2.14: Illustration how color values may be selected from an image database [7]

Recent image-based lighting concepts showed promising improvement in two main areas, in the level of realism as well as the efficiency and calculation speed, leading partially to real-time lighting results. Optimal artificial sky models can significantly improve simulation accuracies with the ability to include light color information and dynamic day-lighting simulations. This can recreate luminance for different dynamic lighting situations and even the most complex cloudy sky conditions, enhancing e.g. models for city planning and architectural purposes. A detailed description is given in the work 'Dynamic day-lighting simulation facility based on image-data' by Bian *et al.* [9].

Kneiphof *et al.* [46] published in 2019 results in connection with the lighting of Microfacet Bi-Directional Reflectance Distribution Functions (BRDFs) with varying iridescence. Iridescence is perceived as '*gradual color changes, depending on the view and illumination direction*' [46], and is an extremely challenging effect to simulate in real-time.

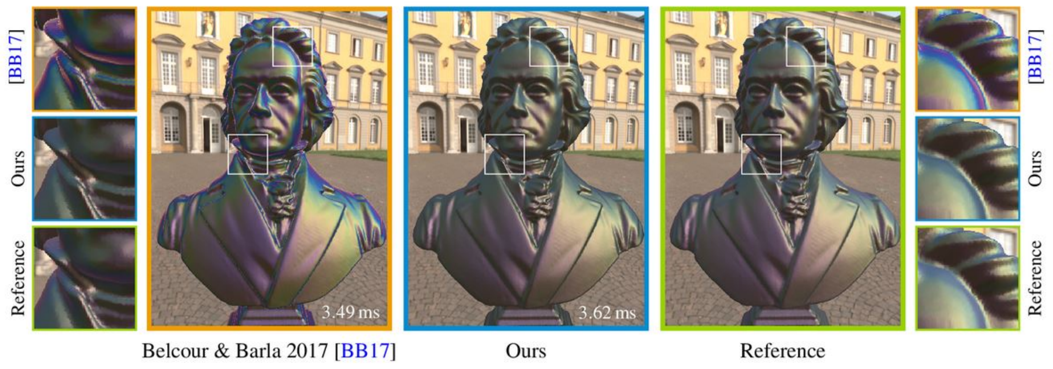


Figure 2.15: Rendering iridescent effects in real-time [46]

2.2.3 Light estimation in MR environments

Currently, automatic light estimators are aiming to replace the conventional IBL method of capturing the lighting, e.g. one module of ARCore (see section 4.4). Nevertheless, the majority of their results is still suboptimal, as the conventional methods lead to more realistic results, especially regarding reflections. These estimators still provide a better light estimate compared to other applications, like Ikea Place. However, a combination of light estimation and IBL is possible and may lead to an enhanced way of lighting.

Nevertheless, promising research for example by Unger *et al.* [91] has been published, who designed and build a real time light probe which supports IBL purposes (Figure 2.16). Their system captures high quality color images of 512x512 pixels with HDR at 25 frames per second.

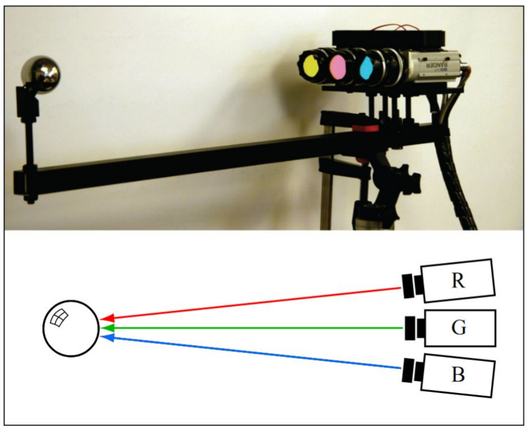


Figure 2.16: RGB capture setup with three cameras aimed at a mirror sphere [91]

Through the reflection in the mirror sphere the researches capture the surrounding scene. Their approach may lead to convincing results, but the presented capture setup is rather oriented for researchers than for users on the consumer level.

As in many related technologies, deep learning also proposes a solution for MR illumination. PointAR [105] by Zhao *et al.* consists of an efficient mathematical model and a compact deep learning model to create a lighting estimation pipeline. Their method published in March 2020 claims to perform well, even on the limited hardware of smartphones, and creates credible results on diffuse objects. Unfortunately, no images with specular reflection were presented and their approach does not apply IBL methods. Park *et al.* [69] aim to calculate correct illumination by recording only a standard photo of a target object without any need for special hardware by utilizing three sequential deep neural networks. Their prediction applies an extensive amount of materials and structures and gradually decreases the dependency on them with the trained networks. Noticeably in Figure 2.17, the specular reflections are rather blurry and lacking detail compared to IBL methods. Nevertheless, the overall performance regarding the color theme is accurate.

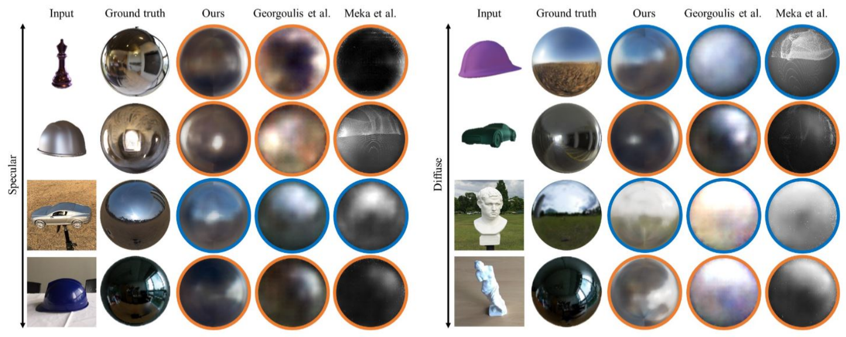


Figure 2.17: Qualitative comparison between rendered results and ground-truth [69]

2.2.4 Feature matching

Feature detection and description is a highly relevant and current topic, as artificial intelligence and robotics paired with the ability to see and understand their environment, create applications and opportunities with endless potential [77].

Scale-invariant feature transform (SIFT) [54], speeded-up robust features (SURF) [6], and KAZE [27] are well-known feature detectors in computer vision with SIFT as the most established one. Lately though with the trend of deep learning, algorithms based on CNNs tend to outperform state of the art computer vision detectors. Learned invariant feature transform (LIFT) [102], SuperPoint [23] and local feature network (LF-NET) [66] are some of the most prominent examples.

Considering matching and registration, Truong *et al.* [88] conducted research connected to the performance of various feature matchers. ORB [76], Binary Robust Invariant Scalable Keypoints (BRISK) [50], Fast Retina Keypoint (FREAK) [2] and the previously mentioned matchers LIFT, SuperPoint, SURF, KAZE, and LF-NET were evaluated in their research, resulting with SuperPoint outperforming others considering feature matching and LIFT showing the best quality of image registration.

One recent paper from June 2020 describes an innovative CNN-based feature detector using so-called 'Greedily Learned Accurate Match Points (GLAM points)', which have been learned in a semi-supervised way. This paper claims to significantly outperform classical feature detectors and even modern state of the art CNN systems [89].

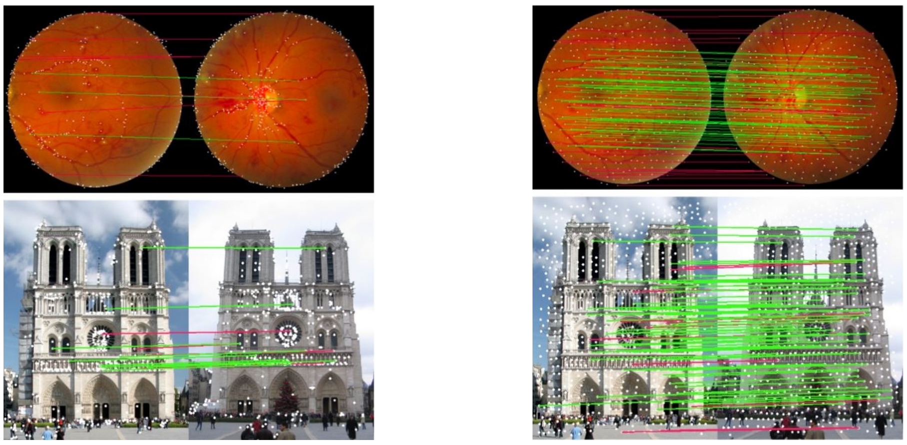

(a) SIFT (b) GLAMpoints

Figure 2.18: Feature points observed by a) SIFT and b) GLAMpoints and obtained matches. Detected features are depicted in white, while red lines false resemble positives and green lines resemble to true positives[89]

As feature matching is often followed by image warping and stitching, some recent research work by Hristova *et al.* [38] features the elimination of color differences that occur during HDR image stitching. This is a common issue in image generation, as seams or abrupt color changes may occur in border regions of image segments. Their displayed work shows impressive results, but unfortunately, their algorithms were not available during the practical process of this thesis. These functions may have improved the quality and outcome of a few obtained images.

Overall, feature extraction and matching will stay an extremely relevant topic as it has become a fixed part in trending technology, leading from autonomous driving to object recognition or even in outer space programs. SIFT was for example used on the Mars Rover by National Aeronautics and Space Administration (NASA) (Figure 2.19).

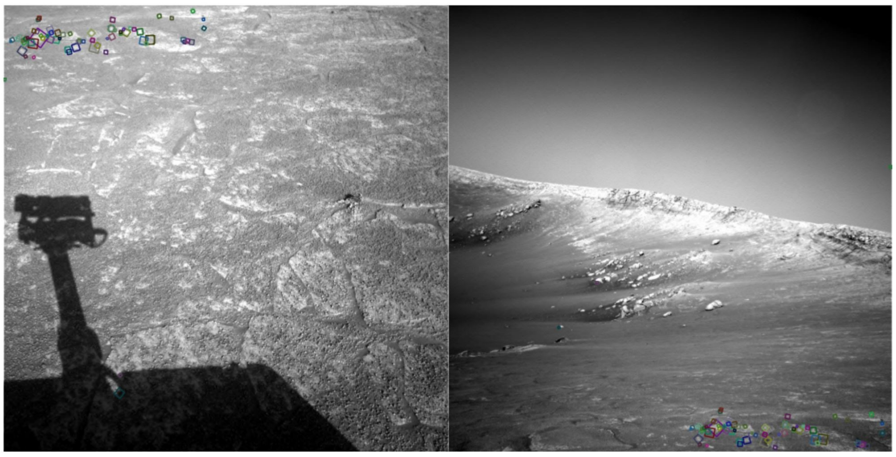


Figure 2.19: NASA Mars Rover images with SIFT feature matches [84]

2.2.5 Optical Flow

Current technologies from 2019 are heading into self- or un-supervised approaches in optical flow estimation. Self-supervised systems aim to learn accurate optical flow estimations reducing the need for pre-training of the system on datasets [53], whereas unsupervised optical flow and stereo depth estimations are currently using CNNs to take advantage of consistencies in the scene, like geometric patterns.

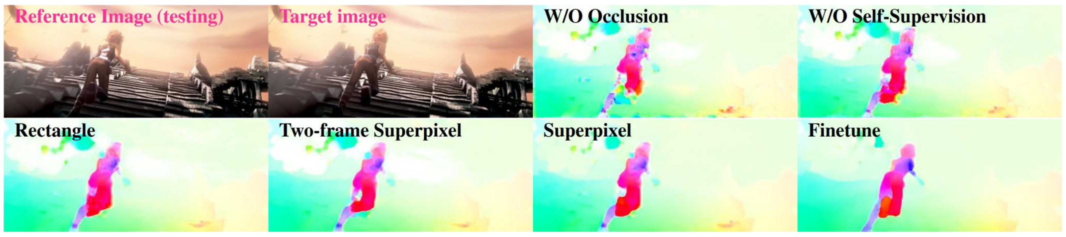


Figure 2.20: SelFlow: Self-Supervised Learning of Optical Flow. Utilizing a self-supervised CNN, occlusions and multi-frame formulation is frequently improved and therefore the optical-flow performance increases [53]

An interesting approach is shown by Wang *et al.* [95]. They use a stereo camera, providing additional information to the CNNs. By showing the scene from two different angles, the system can take advantage of depth information created by the combined information of the stereo images.

Further applications are video denoising, making use of information from adjacent frames to remove noise and other artifacts, or deinterlacing which is '*the process of converting a video taken with alternating fields of even and odd lines to a non-interlaced signal that contains both fields in each frame*' [85]. Overall optical flow is applied in traffic analysis and vehicle tracking or segmentation tasks (Figure 2.21) [65].

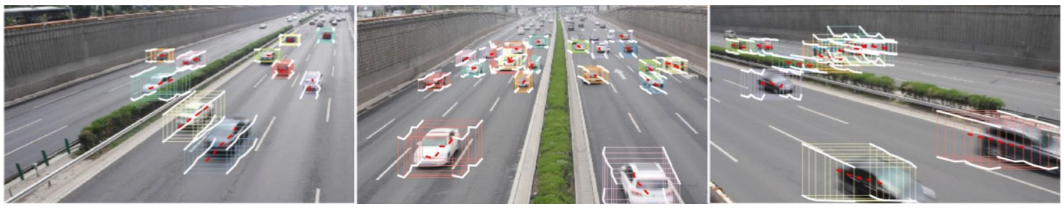


Figure 2.21: Spatial-temporal bounding tubes to model the vehicle's motion [51]

2.2.6 Graph-cut

In current research works, graph cuts are often used as a tool of segmentation or separation of various objects in a scene. Applications reach from image processing for dividing foreground from background pixels [63] or the medical sector like magnetic resonance imaging (MRI) [19] or endoscopic bladder image registration [98].

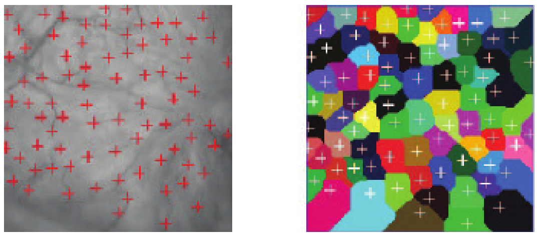


Figure 2.22: Bladder image analysis using Graph-cut [98].

During the research process of graph-cuts, it was noticed that in the last years the basic task of segmentation did not change drastically, as it was discussed in various research papers. On the contrary, the procedure of conduction is differing significantly between papers, as more advanced methods have been developed throughout the years.

For example in the recently published work of Hu *et al.* [39], the system applies multi-threshold segmentation for irregular images, leading to lower error rates in different evaluations and even claims to have a lower run-time than other state of the art image segmentation algorithms. In the cited research the authors present a solution which may even be used on either square and nonsquare input frames, by finding the minimum of a cost function utilizing a so called 'artificial bee colony algorithm'. This calculated value is later applied to find the optimal threshold of the image.

3 System

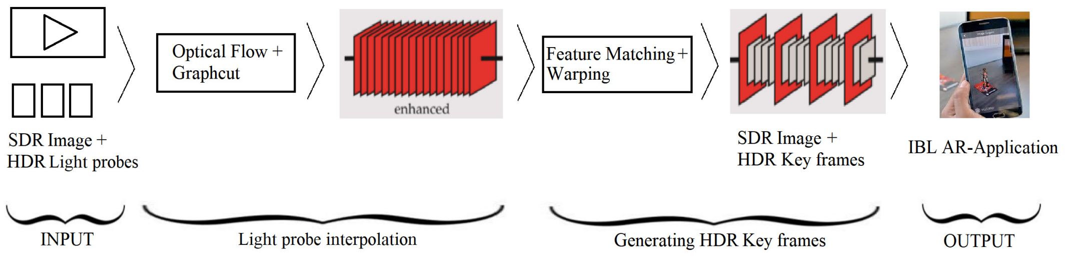


Figure 3.1: Overview of the system structure adapted from [33]

As depicted in Figure 3.1, the presented system involves three parts, which may be interconnected:

- Light probe interpolation
- Generating HDR key frames
- Image-Based Lighting

The main goal of this system is to create an accurate light probe at any position in the recorded scene. To do so there are two different approaches provided, utilizing feature matching and warping, or alternatively optical flow. In any of these options, the scene has to be sampled with HDR images, which may be applied as light probes throughout the scene. Realistically not every random position is sampled and the user might require the application of an exact light probe on an unsampled location.

The user may use optical flow instead of this procedure to calculate an HDR light probe between a pair of adjacent HDR images. In this case, the optical flow is determined between the SDR image and each of the two HDR light probes, generating two flow warped images in the same position.

Following that these images are combined utilizing graph-cut reconstruction and optional artifact handling.

Alternatively, the HDR generation may be conducted with feature matching of the target frame and the nearest HDR light probe. Thereby, the user is required to record a 'standard' image on the target position and following to this the HDR image is warped to that perspective. This process requires additional steps, like HDR image rotation and the conversion into cube maps. This simpler approach has the advantage of a faster execution time, as discussed in section 5.4.

Independently of the procedure, the user creates a fitting HDR light probe on the target position, which may be applied for HDR IBL in an MR environment, like in the case of this thesis - Unity or the OpenGL IBL Viewer.

3.1 Hardware

To capture the required 360° images the provided Samsung Gear 360° camera and a Samsung Galaxy S9 phone were utilized. The provided camera cost at the time of this thesis around 200 euros. Here, the smartphone was applied as a remote control for the camera and as storage of the adjusted images. Moreover, pictures on the camera were directly stored in the fisheye format, whereas on the phone they were stitched together into equirectangular images in the Samsung Gear 360 app.

Typically, in a camera three parameters are controlling the amount of light captured:

- the exposure time
- the aperture - also known as *f-number*
- the sensitivity - also known as *ISO value*

Ideally only the exposure time while recording images should be varied, as the other two parameters also include side effects if they are changed. The aperture affects not only the amount of captured light but as a side effect the depth of field changes as well. On the contrary, the sensitivity also affects the amount of noise in the recordings, which may affect the camera's noise reduction algorithm and might lead to unpredictable results [100].

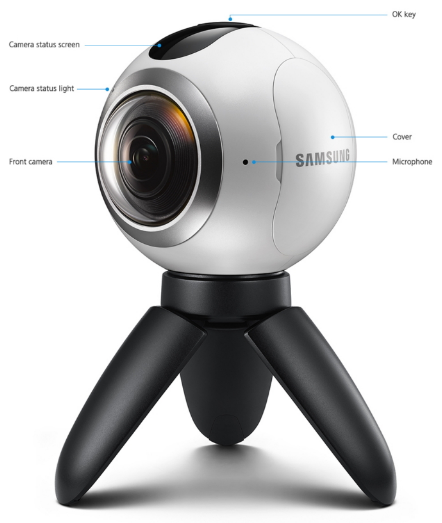


Figure 3.2: Samsung Gear 360 camera schematic [78]

Figure 3.3: Samsung Galaxy S9 smartphone [79]

3.1.1 Capturing process

Before capturing the images with the previously mentioned 360° camera, auto exposure was turned off in the settings as well as the white balance was set to a fixed value, depending on the scenery in the range of 2800 Kelvin (indoor) - 5300 Kelvin (outdoor) [100]. After these steps were checked, several options needed to be adjusted and ensured in the scene, although unfortunately some issues were found after unsuccessful recordings.

A usual room has frequent regions that lack features, making it harder to match those areas successfully later in the matching algorithm. Typical examples are plain walls, doors, curtains, or solid colored surfaces of furniture.

A solution to prevent this is the usage of AR markers which can be generated by various websites or in this case through *Vuforia* target markers [93]. Not only feature-sparse regions affect the matching, but also several similar or identical objects can cause false results in the matching procedure as they could lead to mismatches. Another important property to be aware of is the time of recording the scene, if it contains a light source from outdoors, as the day times change the lighting conditions quite drastically. This has an enormous impact considering matching and the generation of HDR images.

After all those preparations are ensured, seven different exposed images per HDR frame locations and SDR target frames of the scene are recorded. The exposure values reach from -3EV to +3EV in one EV steps. Afterward, the recordings are sent to the Samsung S9 phone, where they are converted to equirectangular images in the process. For every exposure time, the whole panorama should always be covered, otherwise, this may lead to unpleasant blend artifacts in only partly overlapping regions.

3.2 System structure

As the connected terms have been explained, the following sections focus on the different processes in detail. Here, the three parts of the system, which may be interconnected, are explained independently.

3.2.1 Part I - Light probe interpolation

As the starting point of the system, it is necessary to capture seven images with different exposure values on evenly distributed locations in the scene (see subsection 3.1.1). As seen in Figure 3.4 the system uses SDR and HDR frames as input, depicted in the first block of the figure. Following that correspondences between each SDR frame are calculated and used to create two warped HDR images in the position of the target SDR frame. This is executed with the help of the optical flow algorithm. As the last step, as seen in the third block of the figure, the warped images and the

SDR information are blended together and some of the possibly occurring artifacts are handled as well using the aforementioned graph-cut algorithm. As a result, an HDR frame is generated on each position where only an SDR frame existed beforehand. These densely sampled scenes allow the user to apply realistic IBL almost at any position of the scene, leading to a real-world illumination experience.

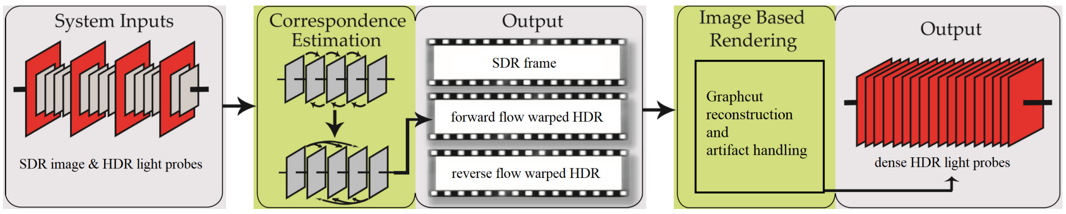


Figure 3.4: Detailed system structure - Light probe interpolation [33]

For this part, the user needs access to a 360° camera that should be connected to a smartphone. By utilizing this hardware it is possible to interpolate the newly recorded SDR frames from the camera between a pair of adjacent HDR light probes. The SDR image provides a practical environment map of the scene, which can be applied in correct localization followed by the interpolation of HDR information with the optical flow and graph-cut algorithms.

During the thesis, several results were obtained using this system leading to various results e.g. highly precise, blurry, or even distorted images. As expected the optimal results were reached with the closest, well-rotated HDR frames in use, whereas suboptimal calculation occurred when the light probes were very distinct from the SDR capture.

The first step after capturing was the calculation of the optical flow between the SDR version of the light probes and the target frame. Thereby, the resulting flow vectors were consecutively applied to the HDR version of the two adjacent light probes: The forward flow to one of them and the backward flow to the other HDR frame. This process generates two HDR frames on the position of the required target frame followed by the calculation of a motion confidence matrix for each warped HDR frame.

Following this process, the next task is to apply a graph-cut algorithm to the two newly generated HDR images with the corresponding SDR frame on the same position as the target image. After refining with additional alpha-beta swapping to lessen probable artifacts, which may have occurred during the calculation, the framework generates an HDR frame with the combined information of the three frames.

3.2.2 Part II - Generating HDR key frames

As overview, like for the other system it is required that at least some HDR light probes exist, which is conducted via exposure fusion (Figure 3.5.a). This is required as the target frame has to be warped from the nearest available HDR image, so optimally they are distributed evenly in the scene (Figure 3.5.b). Up next it has to be decided which light probe is positioned the closest to the recorded SDR target image to ensure the best possible results (Figure 3.5.c). The distance is calculated by Colmap localization in the presented solution. Due to the fact that the equirectangular presentation of an image is heavily distorted the further away the pixels are from the center of the image, it is required to rotate it in several degrees to ensure undistorted image information for the following steps. Additionally, the equirectangular image has to be converted into a cubemap to correctly match the top sides of the images(Figure 3.5.d). Following that the nearest rotated light probe is tone mapped and these images are utilized in feature matching and warping with the target image. The conversion to the SDR domain is required as some algorithms do not support HDR files. Then, the best performing light probes and their transformation matrix are calculated and stored (Figure 3.5.e). To continue the process the HDR image connected to the best performing light probe is selected and the previously stored transformation matrix is applied to it by warping its perspective. As a final step the created warps, which represent single sides of the cubemap have to be converted into an equirectangular image to be utilized as a HDR light probe on the target location.

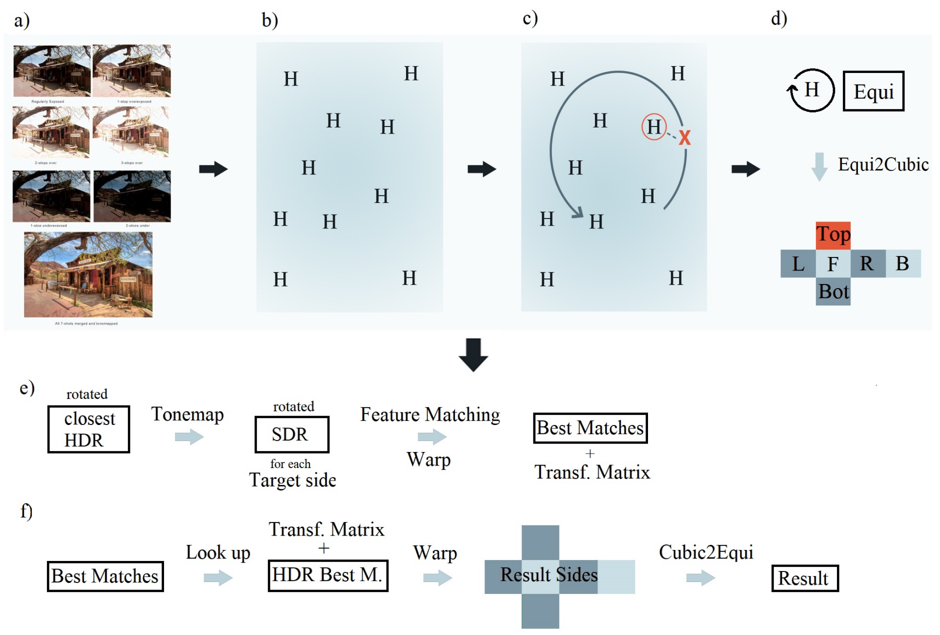


Figure 3.5: Detailed system structure - Generating HDR key frames. Words enclosed by a box represent images. **a)** Exposure fusion **b)** Sparse HDR light probe positions (H) **c)** Choosing closest HDR frame to SDR path (X) **d)** Rotating the nearest HDR frame and conversion into cube map for 'top'side **e)** Tonemapping of the closest rotated HDR frame followed by feature matching and warping to the target sides in the SDR domain leading to finding the best performing matches plus the connected transformation matrix **f)** Utilizing the HDR version of the best matches and applying the previously calculated transformation matrix in the warping of the HDR frame resulting in HDR cube sides which can be transformed to the final equirectangular image

In detail, as the starting point of this part, it is necessary to make sure that HDR light probes are evenly distributed locations in the scene (see subsection 3.1.1). Additionally, one SDR image sequence needs to be recorded, showing the scene. Next, the images are directly transferred from the camera, as well as converted through the Samsung Galaxy S9 with the gear 360° software to obtain both fisheye- and equirectangular images. Then with the help of Picturenaut section 4.5 the recorded exposure brackets are combined and result in one HDR image per recorded image location. The HDR locations are marked as H, while the SDR path is depicted as a line (Figure 3.5.b). After splitting the fisheye images with Photoscape X (see section 4.5) to two different images for the front and the back view of the 360° camera, they are ready to be used for localization purposes. The distance of the frames is calculated by the sparse scene reconstruction functionality performed with the help of the Colmap software (see section 4.5),which served as a development system. Due to its accurate pose reconstructions, the provided information has been serving in the debugging process, as these are vital for the proof of concept. As input for Colmap, the fisheye images of the SDR sequence as well as the zero-EV exposure fisheye images of each of the HDR keypoint locations are used. This is required as for every other frame of the recorded path the spatially closest HDR frame is conducted, as displayed in part c of the system structure, where X resembles the target frame and the circled H the closest HDR frame. As the image location, the average position of the front- and the back-fisheye-image is assumed.

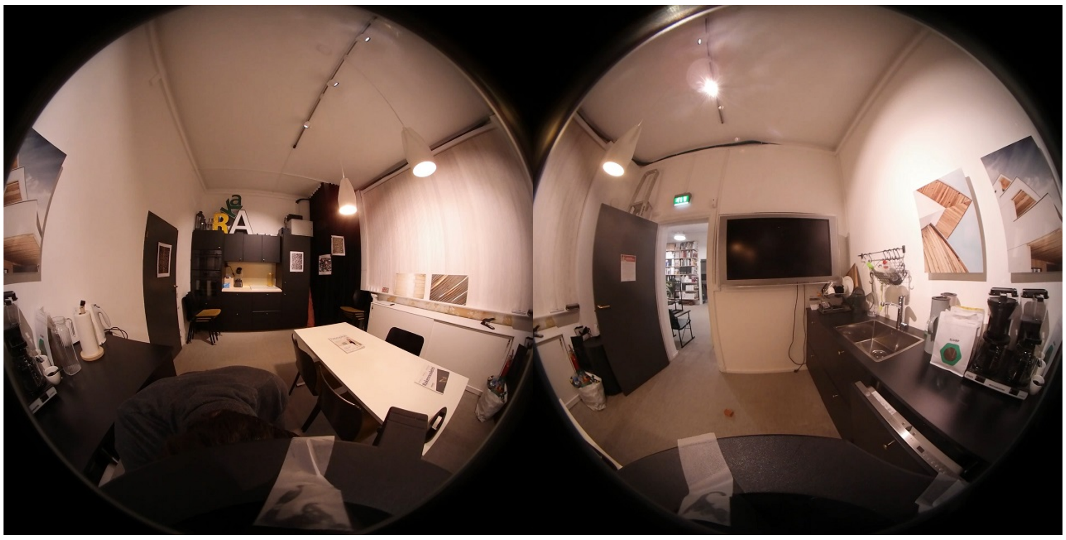


Figure 3.6: Example image directly stored on the 360° camera

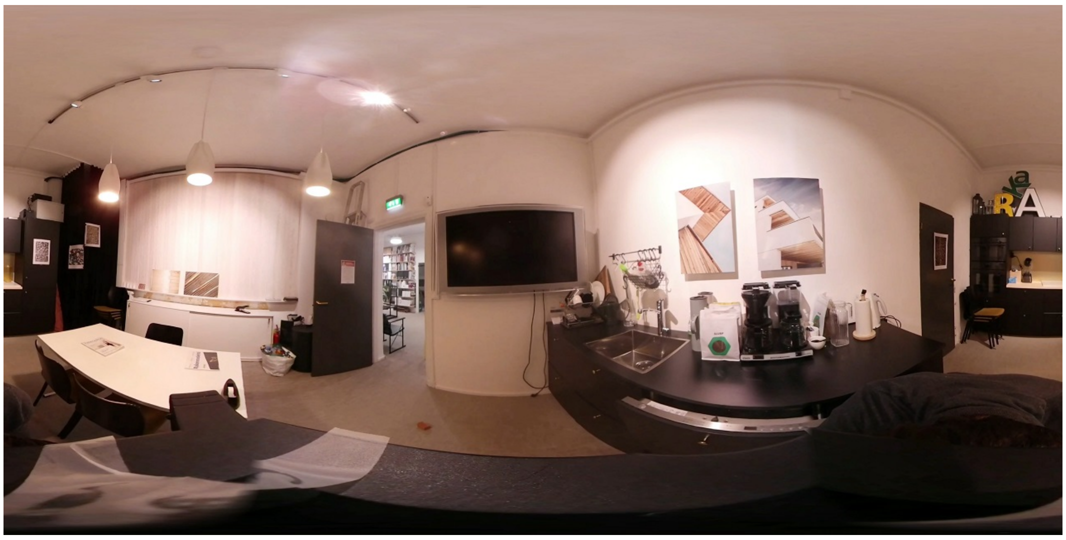


Figure 3.7: Figure 3.6 after conversion to equirectangular

After the target frames and the HDR images have been localized successfully the following steps are a) the rotation in several angles (see section 4.2) of the HDR frame in question and b) the conversion into a cube map using equi2cubic function in Matlab (see section 4.1). The rotation of the nearest light probe is obligatory as the uncentered areas of an equirectangular image are heavily distorted, and a reconstruction of the whole 360° image is intended. Therefore, an undistorted view in every direction of the image is required. The horizontal rotation in various degrees assures that all the sides of the scene occur centered and therefore undistorted in at least one of the rotated panoramas.

The cube map is advantageous as the top and the bottom of the cube map are handled in a special way. The top face of the cube is only matched with the top face of the closest HDR image, as the camera is assumed to be upright in all cases. Opposite, the bottom side is included in the SDR format, taken directly from the SDR version of the target frame. It is assumed that this side does not contain any light sources, as it is rather unlikely that there is any illumination source coming from below. Furthermore, it cannot be matched correctly due to the total differences in the target frame and the combined exposure series. In one situation this side was recorded while carrying the camera, while in the other scene it was placed on top of a pole or tripod. The rest of the individual cube-sides are then matched with the nearest rotated equirectangular HDR images.

Noteworthy, feature matching is performed in the SDR spectrum, therefore the provided and rotated HDR frames have to be tone mapped as displayed in Figure 3.5.e. The resulting image of the SDR image warping is not taken into account for the final image, however, the obtained transformation is stored and later applied to the HDR version of the best performing SDR matches (Figure 3.5.f).

Feature matching This project utilizes a SURF feature detector and a Fast Library for Approximate Nearest Neighbors (FLANN) based feature matcher. Both are part of the Open Source Computer Vision Library (OpenCV) (see section 4.3) as ready-made functions in which the fundamental parameters can be altered by the user. By defintion, '*FLANN contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and high dimensional features*'[3]. It features faster calculations than the alternative matcher for extensive datasets, therefore it was chosen over the *brute-force matcher* (BFMatcher). In the matching process, the image with the most matching unambiguous features is chosen.

The so called Lowe's ratio test [55] was developed in 2004 with the goal of filtering matches allowing only sufficiently different ones.

$$\text{if}\ \ \text{distance1} < \text{distance2} \cdot \text{feature\_distance\_constant then} \ldots$$

Where distance1 is the distance between the chosen keypoint and its best match, and distance2 is the distance between the keypoint and its second-best match. If the second-best match is different enough the match is accepted, as the match is recognized as unique and therefore unambiguous. The feature_distance_variable can be set in the range of 0-1 within the framework, where 1 is the greatest distance and 0 is the strictest variation. In this practical work, values in the range of 0.45-0.7 were chosen due to rejecting features that are too distinct while accepting feature distances that normally occur during the recording process. Setting the feature distance too low may result in only a few matches, sometimes only high feature density regions get matched or areas which are recorded without any kind of interferences or changing angles while recording. Contrary to that a too high feature distance leads to false positives. Thereby, resulting images are told to be the best matching by the algorithm, although the user could clearly tell that the images have only low or no similarity at all.

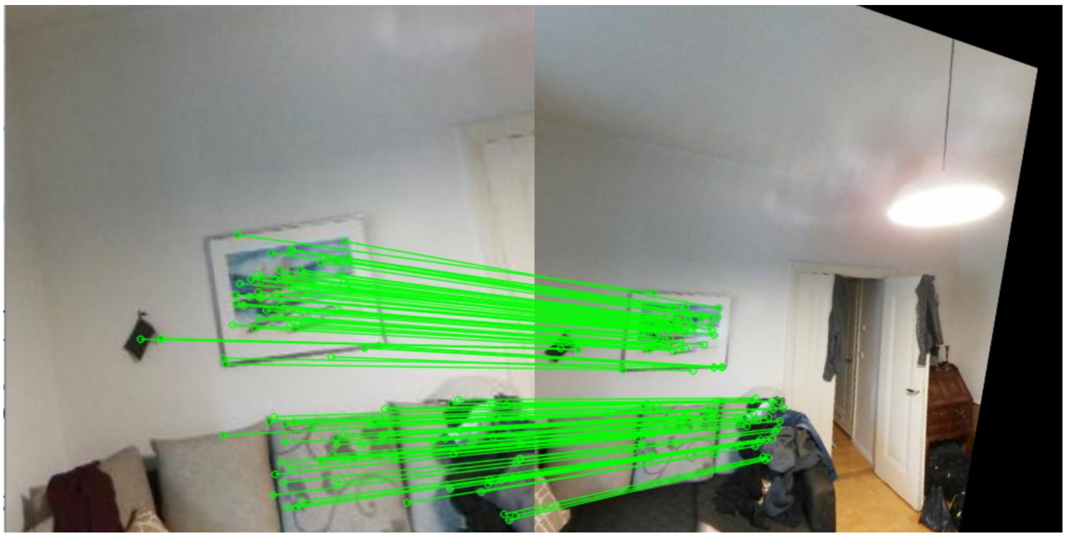


Figure 3.8: Example feature matching

Green lines in Figure 3.8 display feature points in which the matching algorithm decided to be the same in both images. In this example, it is clearly visible that feature sparse areas like white walls or ceilings are non-ideal regions to apply feature matching, as no matching features could be detected or at least classified as identical in both images.

Warping Following the matching of the detected features, the transformation between the tone mapped HDR frame and the target SDR image is calculated by finding the perspective transformation of the image pair. This is referred to as homography and applies their matched feature points. The process of homography is performed in the SDR domain as the functions partly require SDR images. Via computing the homography also a transformation matrix is generated, which may then be applied to the HDR image, containing the same features. The warping process is finalized by warping the perspective from the HDR frame utilizing the generated transformation and ultimately generating the fitting HDR light probe.

To conclude, an example for one frame creation is presented in the simplified Figure 3.9:

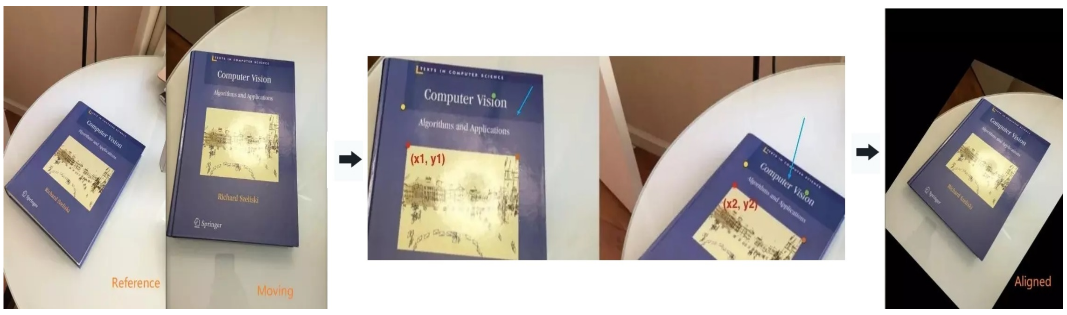


Figure 3.9: Example warp perspective [73]

In the first step, the target frame is shown as 'reference' and 'moving' represents the SDR version of the nearest light probe frame. Next, the corresponding feature points are located in both images and by finding the homography the transformation matrix is calculated as seen in the center part. To finalize the process this transformation matrix has to be applied via perspective warping to the HDR version of the light probe frame to create the correctly transformed light probe in the HDR domain.

3.2.3 Part III - Image-Based Lighting

The illumination model consists of HDR panorama frames that are utilized for IBL and objects with fitting physically based material, representing realistic behavior of the respective real-world object.

Therefore shiny surfaces would be assigned a mainly specular physically based material, whereas matte areas would apply a diffuse material. The majority of illumination originates from the provided panoramas by applying BDRF shading (see subsection 2.1.4), which may be dynamically updated.

Adding lighting provided by light estimators or other light sources is optional. This combined illumination is then applied to the virtual object, which has fitting materials linked to it.

Depending on the system, the HDR panorama is either internally converted to a cube map, as in Unity, or manually converted into a six-sided cube map like in the Open Graphics Library (OpenGL) IBL-Viewer.

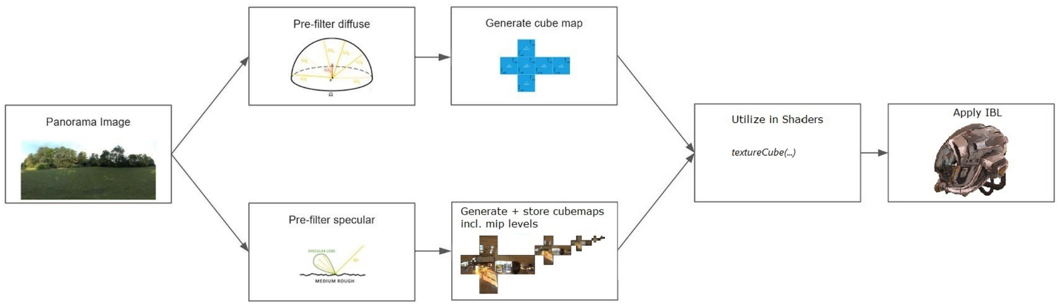


Figure 3.10: IBL Workflow [45]

The illumination deriving from the panoramas is applied utilizing shaders. These shaders process the provided information and determine how to render each pixel, by evaluating the BRDF. Per definition, shaders control the rendering of lightning and shading effects. 'Vertex shaders' affect the position, lighting, and color, whereas 'geometry shaders' generate points, lines, and triangles [40]. Fragment shaders define RGBA (red, green, blue, alpha) colors for all processed pixels. Noteworthy, the illumination of each fragment is calculated individually by combining exposure information from the provided illumination source, BRDF, the model's material, and the model's geometry. This is conducted by interpolating the vertex normals across the entire polygon and then computing ambient, diffuse, and specular lighting for each fragment. Additionally in the scene included light probes support the dynamic change of panoramas at runtime and the fitting illumination from the current cube map.

To achieve a correct localization and scale in the 'real' scene, a dense reconstruction from the recorded SDR image path is necessary. This reconstruction is conducted with the Colmap software at the same time as the sparse reconstruction, needed in finding the closest HDR frame for each SDR frame.

As the virtual objects and scene information has to be linked to the real world, an AR marker is placed below the origin of the image path for initialization.

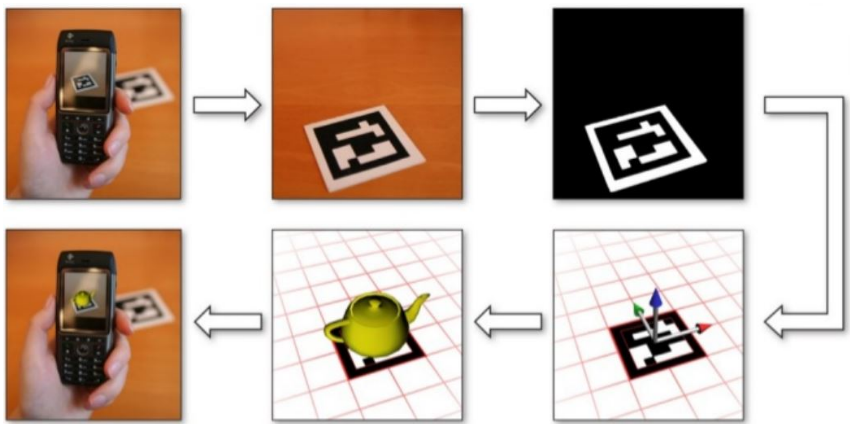


Figure 3.11: AR Marker tracking pipeline [94]

This step is called registration, which is the process of aligning the physical and synthetic world. Here a visual AR marker is utilized for image registration, which is a computer vision task, and correctly determines the camera's position and angle as well as the rotation. Since high precision is vital for achieving immersive AR, the scene has to be correctly set up to match the objects and IBL to reality [10].

4 Implementation

As previously mentioned the presented framework consists of three parts:

- Light probe interpolation to generate additional intermediate light probes by applying optical flow and Graph-Cut
- Generating HDR key frames by feature matching and image warping
- An AR-application which utilizes generated HDR lighting information to illuminate a virtual object

For this thesis, the task of the generation of extra HDR light probes was influenced by several related Python tutorials which focused on feature matching followed by image warping. The majority of examples have applied a variety of ready-made OpenCV functions as part of their source code and likewise, it was implemented as practical approach for this thesis. Therefore, the framework for HDR generation is mainly developed in Python, although at some points some inputs and outputs are used from other Software like Matlab (section 4.1) or Colmap.

The light probe interpolation part is formed by two existing frameworks, adjusted by modifications, extensions, and combined into one. The resulting framework uses a mixture of Matlab code paired with 'mexed' C++ code, leading to the creation of a user-friendly function accessible within Matlab[42]. Therefore changes in the C++ code always have to be 'mexed', meaning creating a usable function which then can be called within Matlab, without the need of using an additional coding environment for running those functions.

Unfortunately, no example code or frameworks were provided by the authors of '*Enhancing and Experiencing Spacetime Resolution with Videos and Stills*' [33]. Therefore, it was needed to refer to the framework of the predecessor of this paper by Bhat *et al.* [8] called '*Using Photographs to Enhance*

Videos of a Static Scene'. This framework seemed to be fitting to the topic of this master's degree project via applying the image-based-rendering algorithm also mentioned by Gupta *et al.*. There were no examples or documentation contained in this framework and after a while of unsuccessful trials it was decided to abandon this framework and an own framework was set up.

Neither optical flow nor graph cuts are a newly invented technology, therefore existing Matlab frameworks that were determined during the initial research phase were combined and extended, as the previously mentioned framework was not functioning and outdated. These frameworks were created by Liu [52] and Bagon [4].

The graph-cut implementation of Bagon [4] heavily relies on the previous research by Boykov *et al.* [103], which developed a technology to make optimal cuts for a given labeling of an image. This means that the image gets fragmented into smooth regions with sharp discontinuities at the border. This is important in this project as a single HDR picture consisting of two warped HDR frames has to be created, which should be fused without any noticeable traces. This is performed by blending the created fragments using a graph-cut algorithm.
As a final result, one framework which runs both tasks with shared input frames and a connected work-space was created. The frames generated in the optical flow function automatically serve as input to the graph cut function. This process may be repeated for several frames which are only available as SDR.

The second part originally consisted of simple feature matching and warping using OpenCV functions but was continuously expanded with functionalities when needed. For example the usage of location information from Colmap, Random sample consensus (RANSAC) [28] functionality, inpainting (section 4.3), tone mapping, and the usage of HDR information. For Python, the Anaconda [41] development environment was applied for changing and extending the source code.

As explained in subsection 3.2.2, for the generation of additional light probes a combination of feature matching and warping is applied to the nearest HDR images and the target frame. The following functions had the biggest impact in this process and were mainly from the OpenCV library:

- xfeatures2d.SURF_create().detectAndCompute() - finds the keypoints and descriptors with SURF
- FlannBasedMatcher().knnMatch(SDR source, SDR target) - matches descriptors, which was followed by choosing the best performing pair
- findHomography(source-feature-points, destination-feature-points,...) - finds the perspective transformation of the image pair and calculates the corresponding transformation matrix
- warpPerspective(HDR light probe, transformation matrix,...) - creates the target image by warping the HDR, utilizing the transformation matrix from the SDR domain

A frequent problem included for instance the brightness values in parts of the code, as they were clamped between 0 - 1 which is the usual way for 'non' HDR images. It was quite challenging to find the locations where the information went missing, but in the final version, all the values remained in the desired range.

Concerning the final application, an IBL viewer was provided by Dipl. Ing. David Mandl from the Institute of Computer Graphics and Vision, Graz, Austria. After the relatively smooth setup of this viewer on Windows with the help of OpenGL, the porting to an Android AR application proved itself as more inconvenient than expected. The majority of OpenGL functions were specially designed to work on Windows systems and had to be converted or rewritten to make use of OpenGL-for Embedded Systems (OpenGL-ES), a mobile version of OpenGL, as well as some file system operations as they are distinct on Android systems. Android implementations are mainly developed in Kotlin or Java, therefore the provided C++ framework had to be converted either to a static library or the functionalities had to be redesigned in Unity. In the last phase of this master's degree project, the Windows OpenGL version was used to generate results for evaluation, whereas a Unity application was created for mobile support.

As mentioned in the system description (see subsection 3.2.3) the HDR panoramas are utilized as the source of lighting with the help of BRDF shading. Every single fragment of the object which should be illuminated is independently evaluated by the BRDF to create fitting and realistic results. It is important that the cube map and the linked shaders are correctly set up and that the materials of the object are physically based, with the result

in adaption to the scene's lighting.
As this project focuses on illumination and lighting, it was vital that all quality settings within Unity concerning lighting, shadows, and similar options were set to the highest level. Otherwise, parts of the system were not functioning as expected, as light probes were low resolution and remained static without the correct settings.

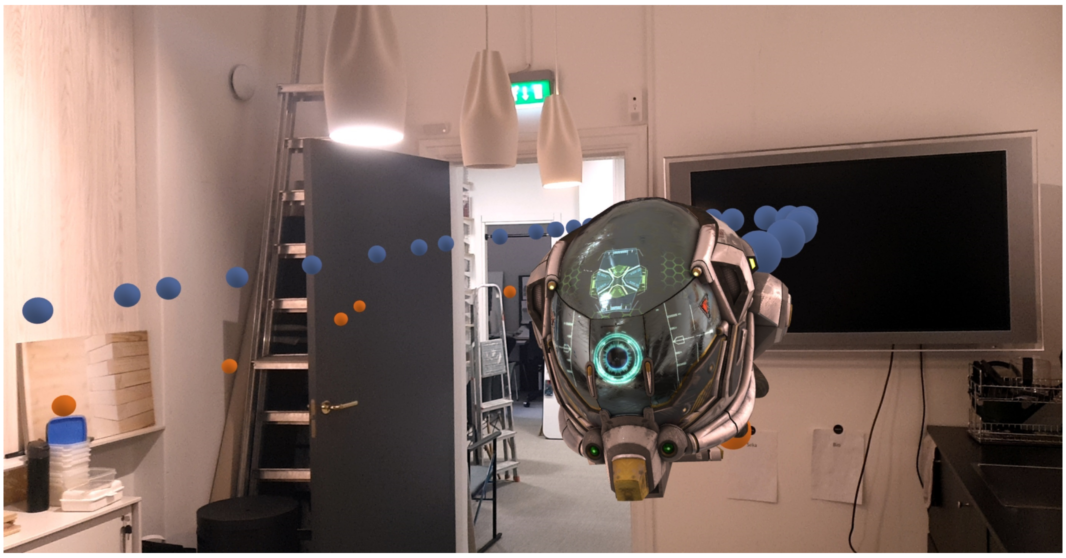


Figure 4.1: Screenshot from created Unity AR application

Each blue sphere represents a image path position, linked to the corresponding HDR panorama and the orange spheres are the positions where the exposure series were recorded. A green sphere is positioned directly above the provided AR marker which serves as a connector of the real and the virtual scene's location (occluded by the helmet in Figure 4.1).

Due to the lack of known scale in the dense reconstruction, the scene's model was scaled utilizing an A3 sized cube within Unity. It was ensured that the reconstructed marker within the scene has the same dimensions as the 'real' A3 sized AR marker. The primary reason for choosing an A3 sized marker was the increased feature-rich area (vs. usual A4). Additionally, the spheres resembling the camera positions were marked as 'children' of the scene' s model within Unity. This guarantees a correct scale and location, as any transformation affecting the 'parent' game object (in this case the scene), is at the same time applied to it's 'children' (the spheres).

The following frameworks and software played a role in the generation of the results of this thesis.

4.1 Matlab

As a core development platform Matlab [42] was decided to be used, as prior experience with this coding environment existed, and the majority of related source code was created within Matlab. It was taken into consideration that the mixing of different coding languages and frameworks can bare difficulties. Therefore provided Matlab frameworks served as the basis for this project, described in the following section 4.1 and section 4.1.
Originally it was unknown that those frameworks rely on previously compiled C++ source code, therefore involuntarily mixing of different coding languages occurred, which luckily did not cause any problems during execution. For C++ the Visual Studio [59] development environment was applied for changing and extending the source code.

Matlab HDRI Toolbox The hdrIMwrite() and hdrIMread() functions were part of an HDRI framework to the book '*Advanced High Dynamic Range Imaging: Theory and Practice*' by Banterle *et al.* [5], and extended the standard Matlab functions by the capability of reading different '.HDR' formats.
In principle, this so-called 'Toolbox' has a way broader bandwidth of features, but those two functions are the only ones which were vital during this project, aside from the image comparison functions to compare the results. This Toolbox is explained as an extra chapter of the previously mentioned book '*Advanced High Dynamic Range Imaging*' [5].

Code by C. Liu et al. The project by Liu *et al.* [52] consists of 'mexed' code written in C++ language. The provided source code was applied and modified in this thesis during the part which dealt with optical flow. Originally the framework calculated and displayed the optical flow between two frames.

During this project, this framework was extended by the functionality of the summation of calculated optical flows, and the application of optical flow vectors onto other frames, in this case, HDR input frames. At the end of the reworking process, the framework was altered to make use of the given SDR frames and calculate the flow between these frames. Additionally, the two adjacent HDR frames are used as input for the function which applies the calculated flow onto the HDR images, to generate warped images in the position of the existing SDR frames.

Code by Bagon et al. The project by Bagon *et al.* [4][12][47][103] consists of 'mexed' code written in C++ language. This source code contained the functionality of graph-cuts within Matlab. The given framework was altered to function with the created warped HDR frames, the calculated motion confidence from the optical flow part, and the given SDR images. At the end of the rework, this adjusted part of the code generated the final result with the inputs from the warped images as well as the frames generated by the optical flow part.

Code by Phan The provided functions *equi2cubic* [70] and *cubic2equi* [71] included the functionality to use an equirectangular version of a scene and create six cube faces that represent the scene and vice versa.
An example of an equirectangular image is seen in Figure 3.7. This was required in several parts of the project, for example:

- for extraction of single sides of the scene, as for example the top side gets only matched with other top cube faces.
- to insert the SDR bottom face into equirectangular images
- to handle a panoramic scene without using the full input frames at once

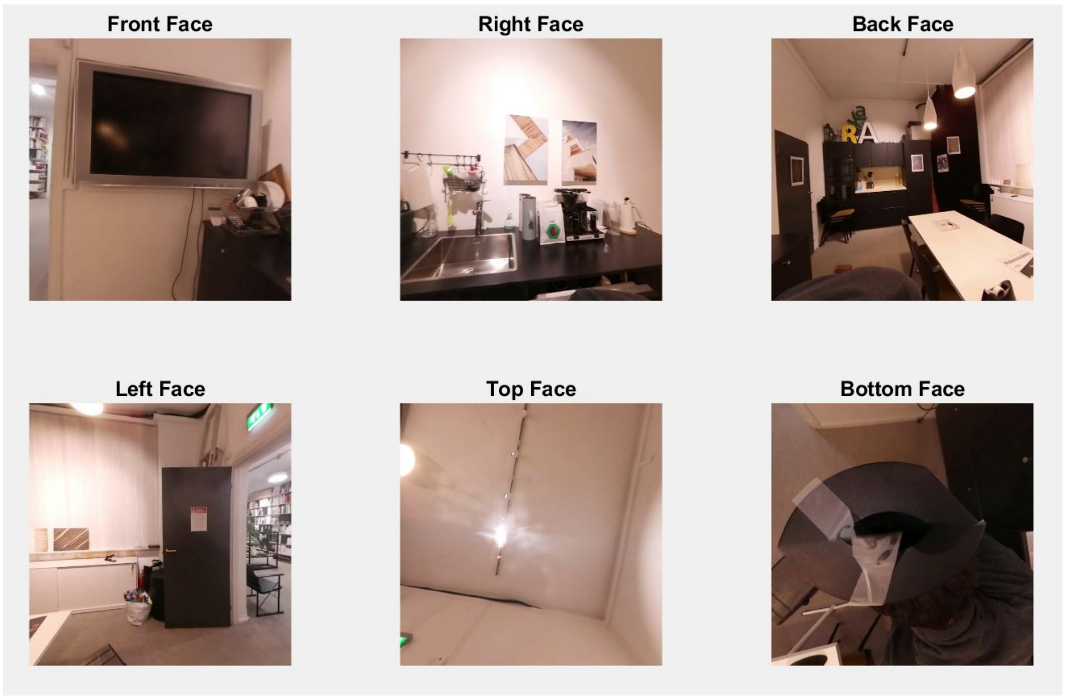


Figure 4.2: Equirectangular image transformed into cube-faces

4.2 C++

IBL Viewer The IBL C++ Framework with OpenGL functionality was provided by Dipl.-Ing. David Mandl from the Institute of Computer Graphics and Vision, Graz, Austria (ICG). Originally the framework supported rendering of an HDR image onto a skybox and using the image information to illuminate a virtual 3D model.
For this thesis, the functionality was extended by dynamic loading and displaying of video frames on the skybox, rather than a single HDR image, as well as using dynamic HDR lighting information to illuminate a virtual object. In the final version of the program, the SDR image sequence of a scene can be played on the skybox and at the same time an HDR light probes provide the IBL. Ultimately the project needed to be recreated in

Unity, as Android did not support the common OpenGL functions like on a computer operating system. Nevertheless, it provided a working solution on Windows 10 environments and it was utilized to generate the results used for comparison in chapter 5.

Equirectangular rotate This program is which is coded in C++ language is able to rotate equirectangular images with Euler angles using OpenCV functions. The user has the option to specify different angles for the roll, pitch, and yaw rotation. It was originally developed by ChoYG [15] and adapted as part of this thesis to handle HDR values and the storing of HDR images.

4.3 Python

OpenCV As inspiration, a sample feature matcher in Python using OpenCV was used as a base for the feature matching task in this project [3]. Functions from the OpenCV library like cv2.FlannBasedMatcher() or cv2.xfeatures2d.SURF_create() were ready-made parts of the source code. Throughout the project, several OpenCV functions were utilized, which simplified the process of feature matching and image warping.

HDRI Inpainting HDRI Inpainting is an inpainting framework which implements several classic image inpainting methods in Python and Matlab. These algorithms complete and supplement the book '*Digital inpainting based on the Mumford-Shah-Euler image model*' [25].
The included Mumford-Shah Inpainting with Ambrosio-Tortorelli approximation was chosen as it generated the best fitting results in the terms of credibility. The only adaptation to the source code was the change to their image writing process to handle '.HDR' input images as well [68].

4.4 Unity

Unity is a game engine developed by Unity Technologies and supports the creation of software for a variety of use cases, e.g. three-dimensional, two-dimensional, AR, VR. What separates Unity from other game engines is that many functionalities are already included in the software, like the physical behavior of objects or different light sources. The primary coding language for Unity is C#.

ARCore ARCore is a platform for building augmented reality experiences developed by Google [30]. Their sample project *Hello-AR-C* served as a starting point to develop an AR app to display the resulting IBL in an AR scenario. In subsection 5.6.2, their application served to create comparative results, as it is a commonly used platform featuring automatic light estimation.

4.5 Software

Shotcut This program was used to extract the single frames from the recorded and provided (HDR) videos for evaluation, as this framework currently requires frames as input rather than a video file [83]. Additionally, the Shotcut software was applied as a video editing software to edit the resulting videos.

Picturenaut Picturenaut was used to generate HDR Images by combining several bracketed images with different exposure values. Namely -3 EV, -2 EV, -1 EV, 0 EV, 1 EV, 2 EV and 3 EV. This process was needed, as the provided 360° camera was unable to capture HDR images by itself. During the HDR creation, it was possible to assign each image an exposure value, and additionally, it was possible to enable anti-ghosting and anti-blur [36].

PhotoScape X PhotoScape X is a photo editing software that provides features like a photo viewer, batch photo editing, and more [86]. In this project, it was used to split the fisheye images provided by the camera (Figure 3.6), into two separate fisheye images in a bulk. This step is necessary as Colmap needs extra fisheye images for the front and the back view in its image localization algorithm, while the original images contained the front-facing fisheye camera view, and the back facing view on a single image.

Colmap Colmap is best described as a SFM and Multi-View Stereo pipeline. It offers a wide range of features for image reconstruction [81][82]. This software enabled the accurate localization of the HDR key frames and the SDR frames in the given scene to find the nearest key frame to each target frame. The images had to be provided as SDR fisheye images as equirectangular image types were not supported at the time of this publication. For the needed purposes a sparse scene reconstruction was sufficient while localizing the frames. Through the utilization of a quaternion plus a translation vector, the pose of an input image may be reconstructed. The coordinate system of the input image is oriented that the X-axis points to the right, the Y-axis downwards, and the Z-axis to the front [81][82].

Luminance HDR This program was used to tone-map the given and generated HDR images to SDR images with the usage of various tone mappers, although mainly the *Reinhard05* tone mapper was applied. Additionally, the program was used to evaluate the given and generated HDR images with bare eyes, as the standard image viewer is not capable of displaying these frames. For a quick evaluation if an image is close to a fitting solution this method is sufficient and an in-depth comparison was used as the result became close to the reference output. An included slider in this program allows the user to change the brightness in the given image, and the user may use it to make previous regions in the image more visible/detailed [56].

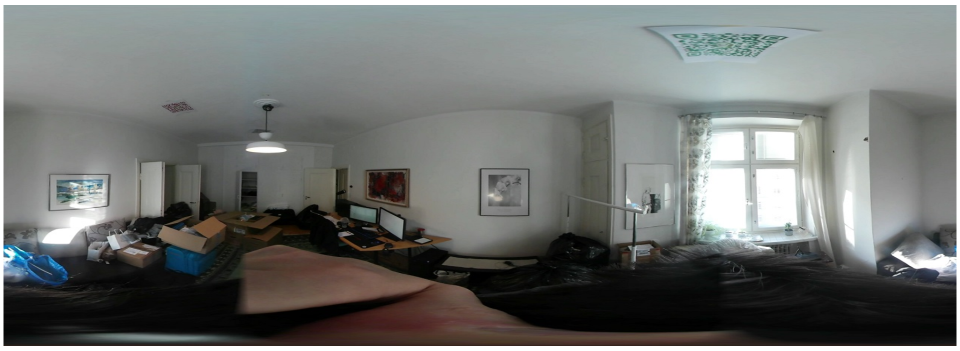
5 Results

As the performance of this work is affected by several factors e.g. amount of HDR frames, scenery, or model, this section compares the results in different settings.

The provided scenes feature a set of different lighting conditions. On the contrary, the chosen models display a range of different available materials, which feature distinct lighting properties. The amount of recorded HDR frames affects the run-time of the algorithm as each of them is checked for fitting features. As the IBL can be calculated before usage, the execution time of the algorithm has minor impact. Depending on the number of total images the result may improve if additional images are added which include better fitting matches of features. Some information that is generated during each execution step can be reused and so the total time per frame-calculation is reduced if several results are generated at the same execution cycle. Examples for reusable information are extracted features, tone mapped images, rotated- and to cube map converted panoramas. Due to varying conditions image matching, -warping and, -stitching have to be computed repeatedly. The majority of information can be computed offline and may be stored without an impact on the performance. Thereby, the whole HDR frame generation process is precomputed, and also the scene's structure is predefined. Only the localization of the AR marker, symbolizing the initial position and rotation, and the correct illumination has to be computed during runtime.

5.0.1 Recorded scenes

(a) Kitchen-scene: Only artificial light


(b) Flat-scene: Natural- and artificial light

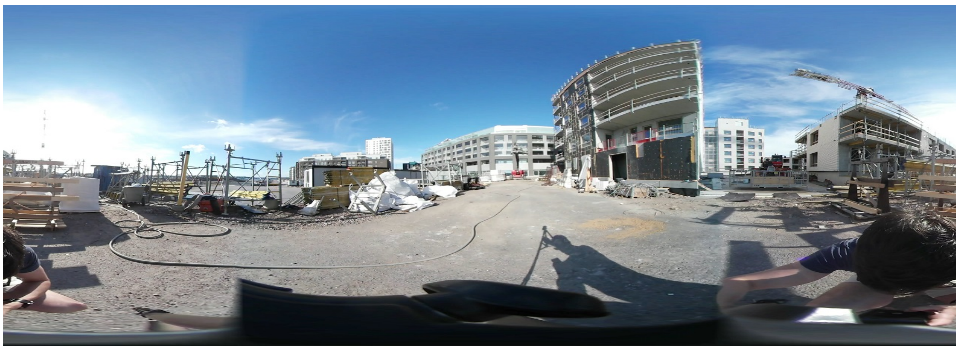

(c) Outdoor-scene: Only natural light

Figure 5.1: Different recorded scenes

5.1 Qualitative results

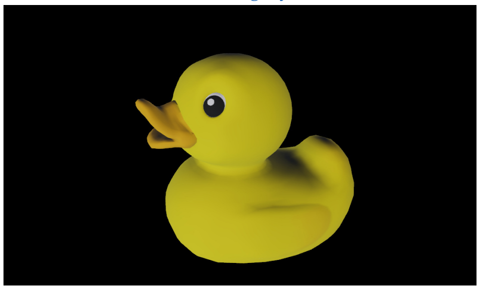
In the following section qualitative image results are presented comparing:

- Scenes
- Models
- SDR versus HDR
- Created versus 'real' images

The selected scenes contain different light setups, artificial light, partly natural- and artificial light, and only natural light (subsection 5.0.1). Intentionally, the chosen models differ in their physical illumination behavior. Highly reflective, diffuse, partly diffuse, and specular and real-world objects are presented.

(a) Mirror Shield: Highly reflective [32]

(b) Corset: Diffuse + specular details [45]


(c) Duck: Diffuse Object [45]

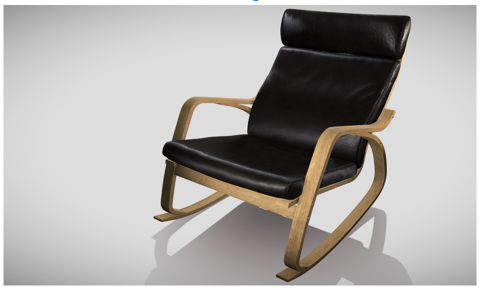

(d) Ikea chair Poäng: Real-life Object [13]

Figure 5.2: GLTF Models

These models were provided in the *GLTF*-format [45], which were loaded into Unity as well as the IBL Viewer. For Unity, a special importer plugin was required [16].

5.1.1 IBL Results on different Models: Kitchen Scene

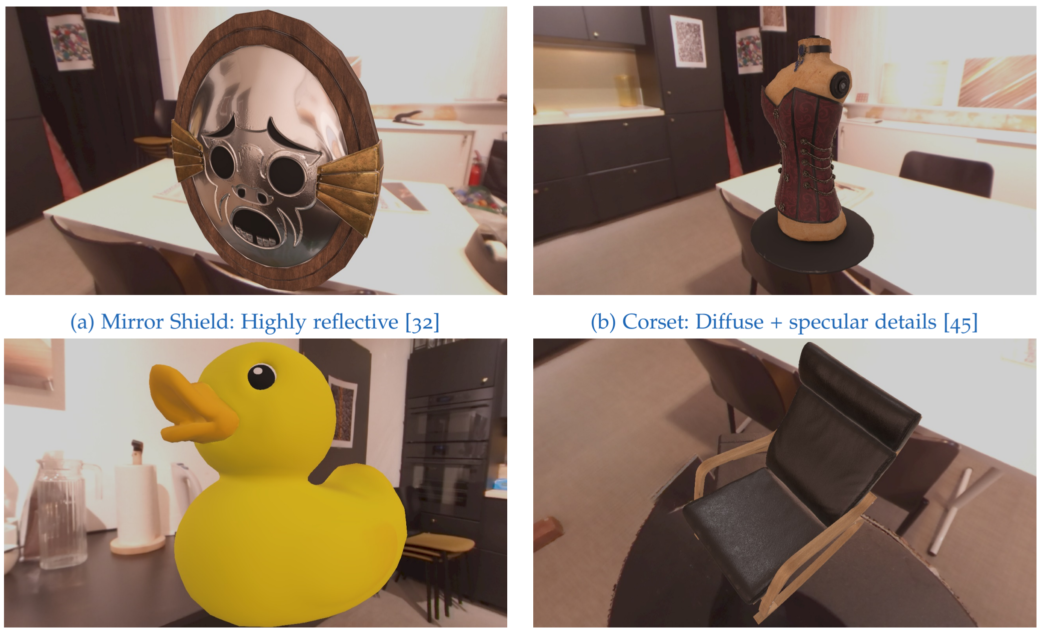
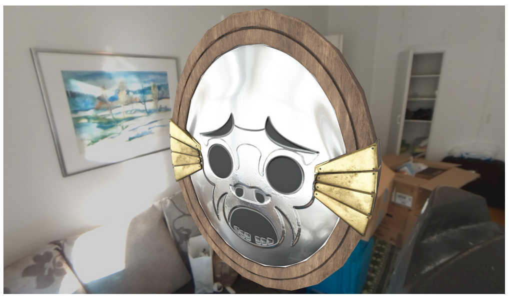
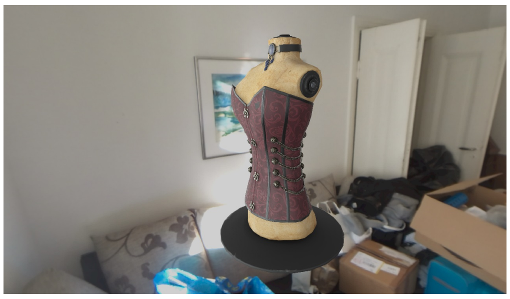

(a) Mirror Shield: Highly reflective [32]
(b) Corset: Diffuse + specular details [45]
(c) Duck: Diffuse Object [45]
(d) Ikea chair Poäng: Real-life Object [13]

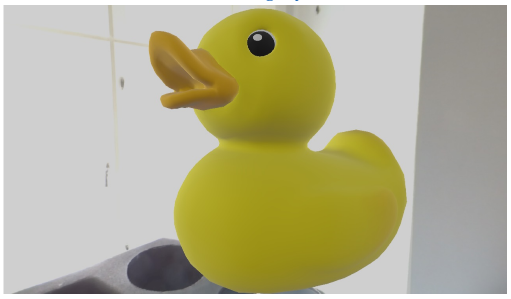
Figure 5.3: IBL results on different Models: Kitchen Scene (tonemapped)

The kitchen scene displays a slightly warmer white balance originating from several ceiling lamps and is generating credible results. All illumination in the scene is artificial lighting, as the room does not contain any windows and therefore no impact from other light sources is available, which is an advantage of this sealed-off location as some variable factors are reduced. The most remarkable impact is seen in the shield (Figure 5.3.a), as it reflects details from the scene, whereas the other three objects appear marginally redder. Noticeably, a duplicate door is reflected in the shield, which is an example of one of the errors that may occur during the HDR generation and is discussed in section 5.5. In the other objects, this error is unnoticeable due to a lower specularity in their materials. In the leather chair (Figure 5.3.d), it is noticeable that the light sources are positioned above the model, therefore the seat is brighter illuminated than the back of the chair.

5.1.2 IBL Results on different Models: Flat Scene


(a) Mirror Shield: Highly reflective [32]


(b) Corset: Diffuse + specular details [45]


(c) Duck: Diffuse Object [45]

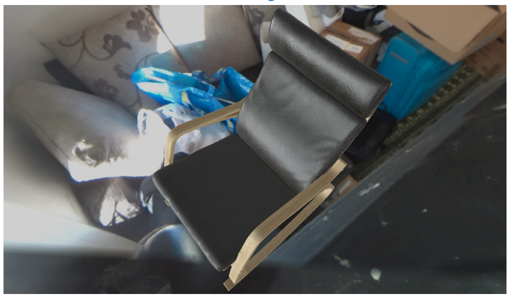
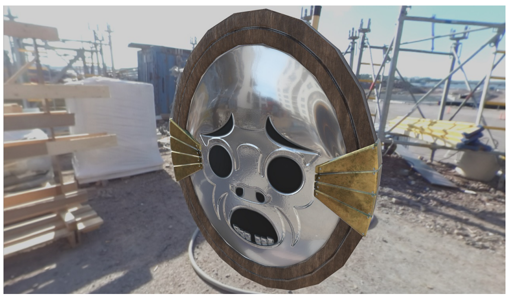
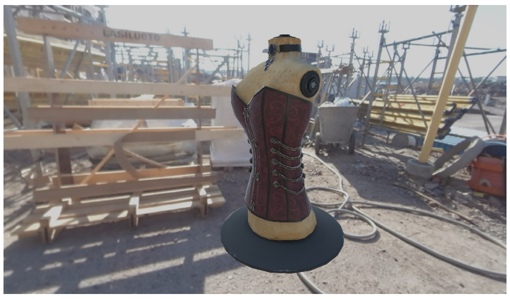

(d) Ikea chair Poäng: Real-life Object [13]

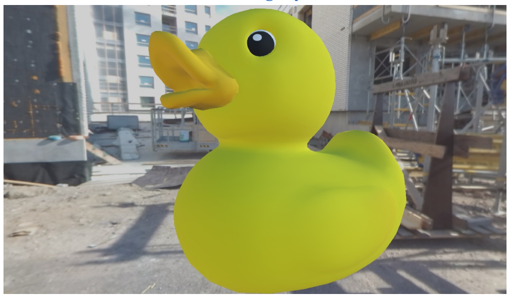
Figure 5.4: IBL results on different Models: Flat Scene (tonemapped)

In the flat scene, the light originating from the window heavily impacts the different models. The scene was recorded on a sunny day and no shades or curtains occluded the windows. All available lamps in the room were switched on to add extra light sources to the scene. Here, the specular parts of the objects are brightly illuminated which are facing the windows, the back of the chair (Figure 5.4.d) is shiny and even the tiny specular details on the corset (Figure 5.4.c) are noticeably bright. In comparison to that the diffuse duck (Figure 5.4.c) is generally lighter than in the kitchen scene, depicting the existing impact on purely diffuse objects. On the contrary, the indoor lamp has hardly any influence on the IBL, as the sun rays from the window have the highest luminance and therefore the highest effect.

5.1.3 IBL Results on different Models: Outdoor Scene


(a) Mirror Shield: Highly reflective [32]


(b) Corset: Diffuse + specular details [45]


(c) Duck: Diffuse Object [45]

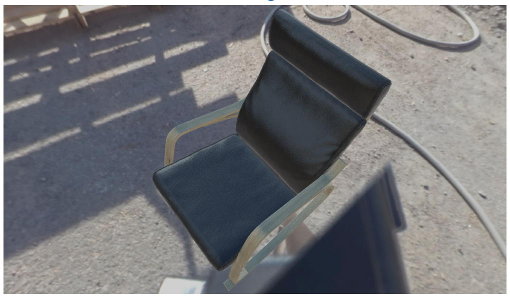
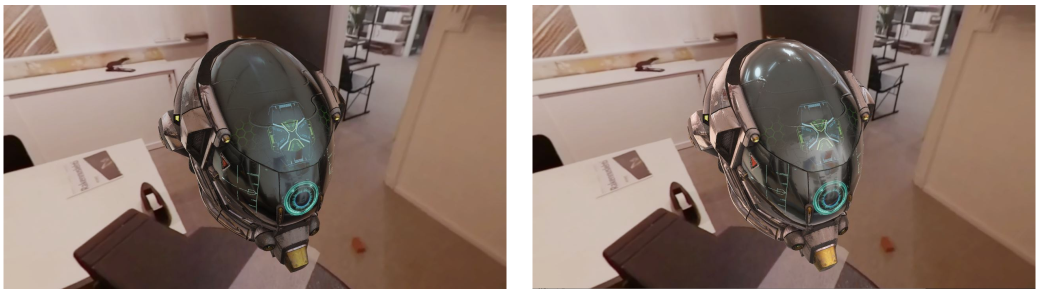

(d) Ikea chair Poäng: Real-life Object [13]

Figure 5.5: IBL results on different Models: Outdoor Scene (tonemapped)

Outdoors the scene is dominated by high illumination values, as the sun serves as the main light source. This construction site was recorded on a sunny summer day without any clouds in the sky, which increases the impact of the sun during IBL. In the shield (Figure 5.5.a), nearby objects are reflected credibly and the already bright appearing models are even lighter than in the flat scene, where the sun rays only shine through the window. The buttons and small details on the corset model (Figure 5.5.b) appear shiny and the diffuse wine red parts have high luminance. Due to the bright illumination, the rubber duck (Figure 5.5.c) model seems like it is neon-colored and the leather chair (Figure 5.5.d) resembles real leather when it is hit by sun rays. Overall the scene's lighting is serving a realistic result mimicking the illumination of a sunny day.

5.1.4 SDR illumination versus HDR illumination

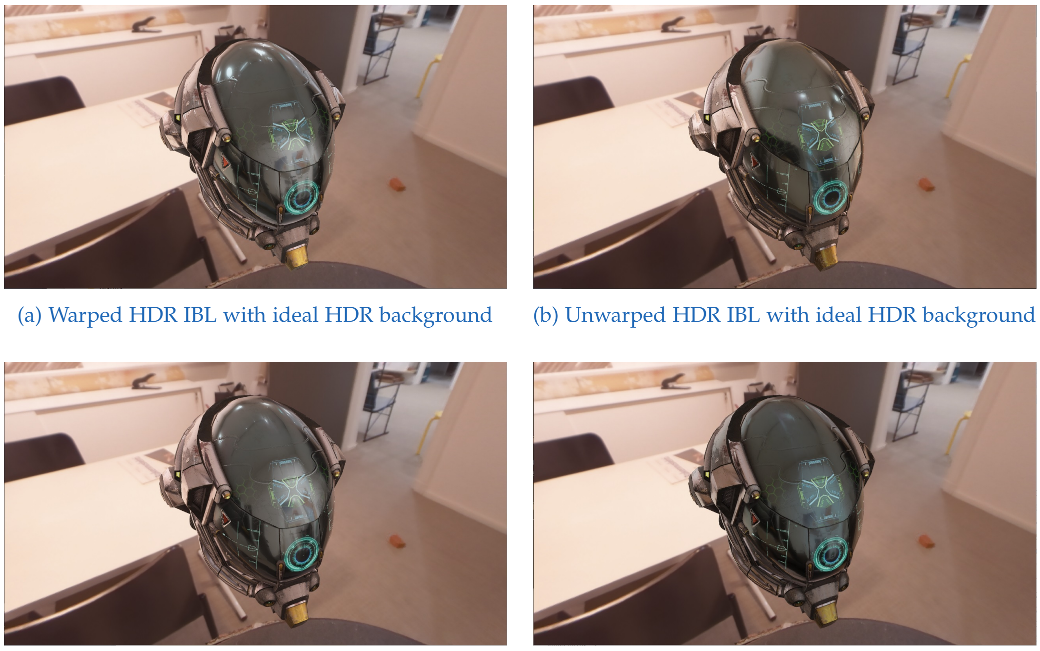


(a) SDR IBL with SDR background (b) HDR IBL with SDR background

Figure 5.6: SDR illumination versus HDR illumination (tonemapped), Damaged Helmet model [45]

For comparison, both models were positioned identically and the SDR frame which was applied in Figure 5.6.a for IBL served as background in both of the images. The difference in illumination between an object with SDR to HDR IBL is best described with more dull reflections and lacking details. In the specular parts of both helmets, the nearby paintings and other objects are visible, as both versions apply IBL. Here, the biggest impact is seen on the hardly existing reflections of light sources, as the available brightness range in SDR is simply not capable of displaying values beyond 255. As previously mentioned also only fixed-point values are available for SDR images, while HDR images consist of floating-point numbers. In Figure 5.6.a, the whole helmet model consists of more or less a more matte colored palette. In mainly diffuse areas the impact is not as severe as in stronger reflective parts but still noticeably greyer, lacking highlights.

5.1.5 HDR warping vs. un-warped HDR vs. HDR ground-truth vs. SDR ground-truth

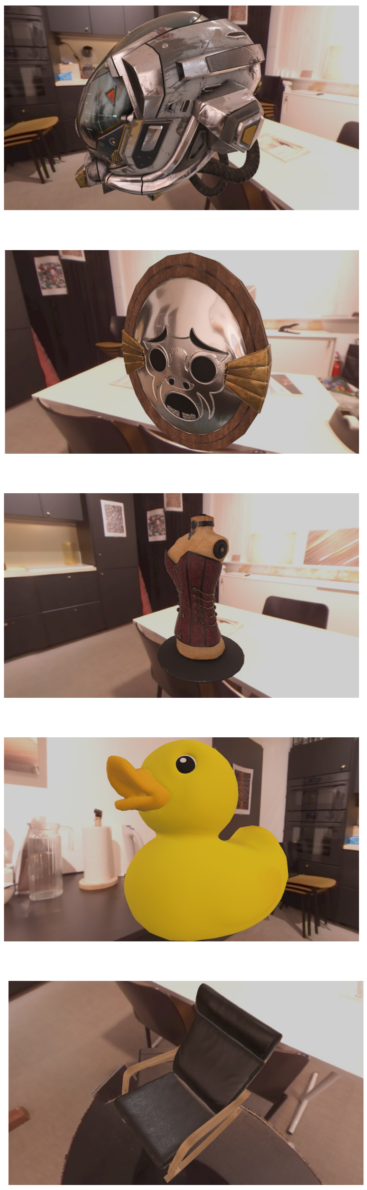

(a) Warped HDR IBL with ideal HDR background (b) Unwarped HDR IBL with ideal HDR background

(c) HDR ground-truth IBL with ideal HDR background (d) SDR ground-truth IBL with ideal HDR background

Figure 5.7: HDR warping vs. HDR lightprobe vs. HDR ground-truth vs. SDR ground-truth(tonemapped)

Here, a comparison between all utilized scenes for IBL is presented. In Figure 5.7.a and (b), the spatially closest HDR light probe and the warped HDR from the created system are compared. In both images the dynamic range and the warmth of the white balance of the scene is perceived in an identical manner. Nevertheless, the result is lacking realism especially in the reflective parts of the model. Objects located in deviant directions in the real scene are present in the specular reflection leading to poor performance results of credibility. The target illumination in the HDR domain of the identical location is presented in Figure 5.7.c and as SDR in Figure 5.7.d. Thereby, differences to the original scene appear obvious. The differences between HDR and SDR illumination have been discussed in the previous section (subsection 5.1.4).

5.1.6 IBL Results: Created versus 'real' HDR Image


(a) IBL from calculated HDR

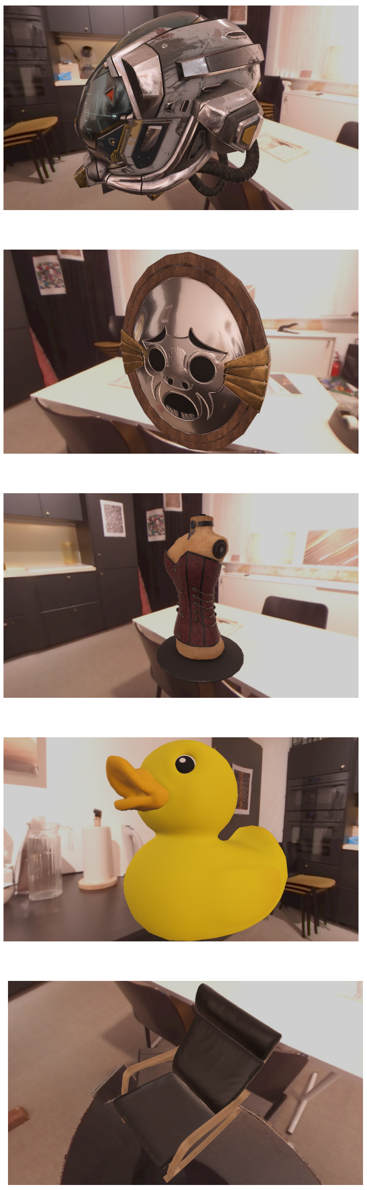
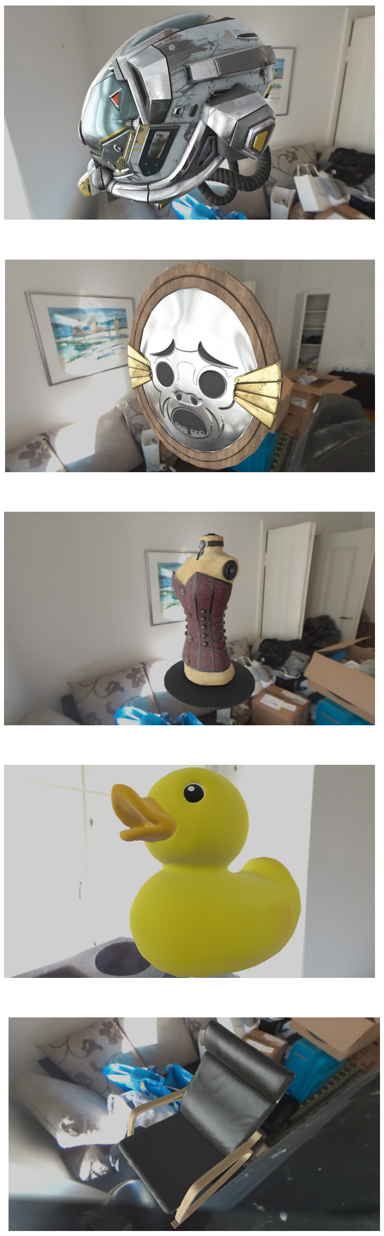

(b) IBL from recorded HDR

Figure 5.8: IBL results: Created versus 'real' HDR-Image (tonemapped) - Kitchen scene


(a) IBL from calculated HDR

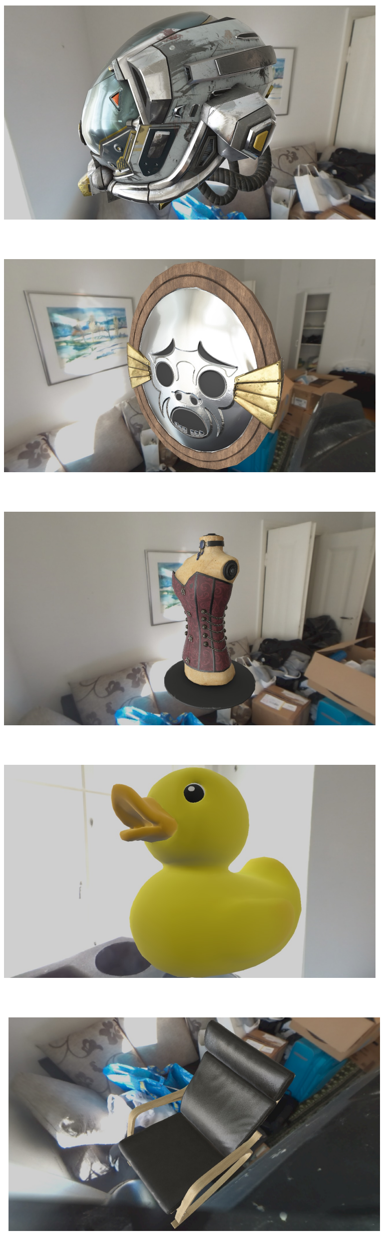
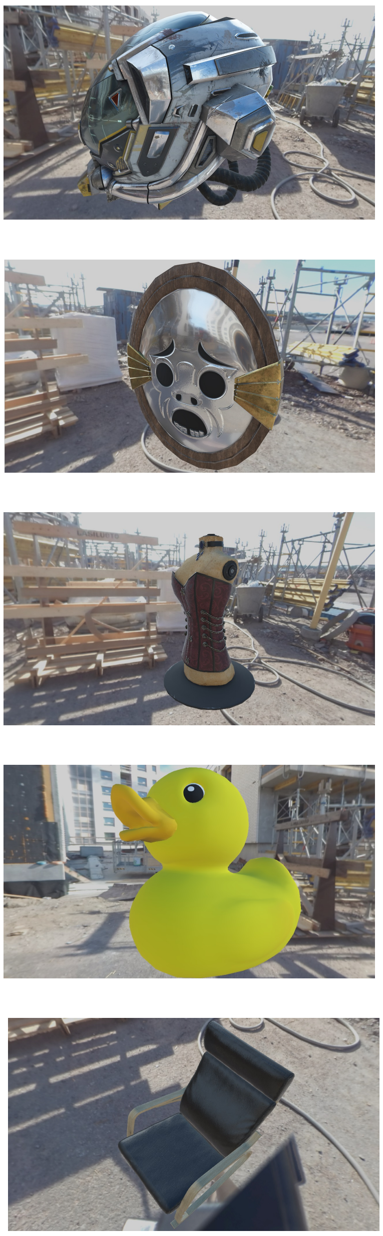

(b) IBL from recorded HDR

Figure 5.9: IBL results: Created versus 'real' HDR-Image (tonemapped) - Flat Scene


(a) IBL from calculated HDR

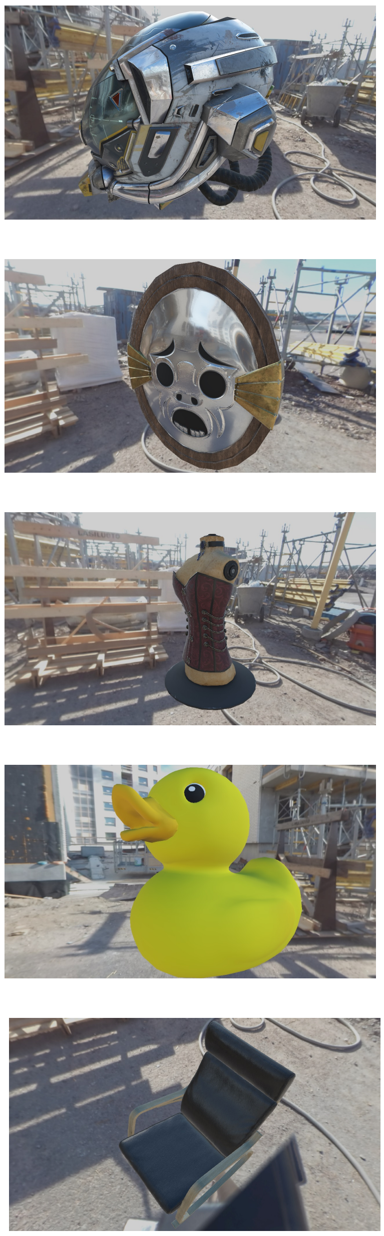
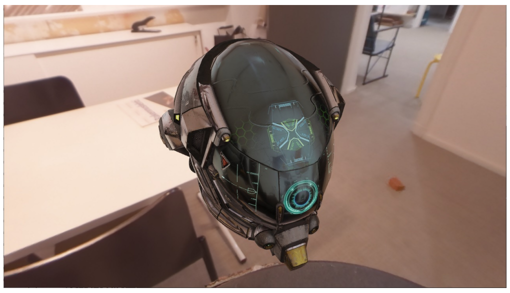
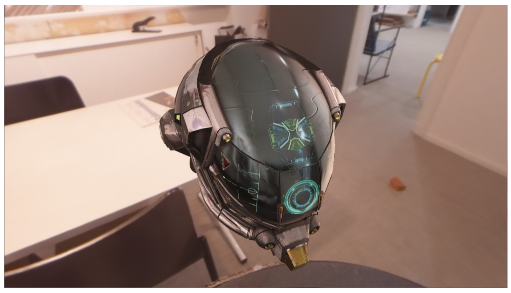

(b) IBL from recorded HDR

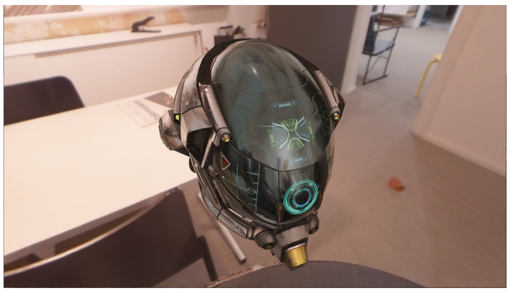
Figure 5.10: IBL results: Created versus 'real' HDR-Image (tonemapped) - Outdoor Scene

Here, the IBL of a generated HDR frame is compared to the IBL of a recorded HDR frame on an almost identical position in the scenery. As background, the same HDR image is used in both cases, whereas it is only applied as a source of illumination in the (b)-versions above. The comparison shows promising results as the images are hardly distinguishable. Minor details like a less dominant reflection of the marker or a slightly bigger reflection of the window are noticeable. Nevertheless, high levels of credibility concerning illumination and reflections are present in the generated IBL.

5.1.7 Nearest HDR IBL vs. Optical flow results vs. SDR ground-truth


(a) SDR ground-truth IBL


(b) Nearest HDR light probe


(c) Optical flow HDR IBL using available light probes

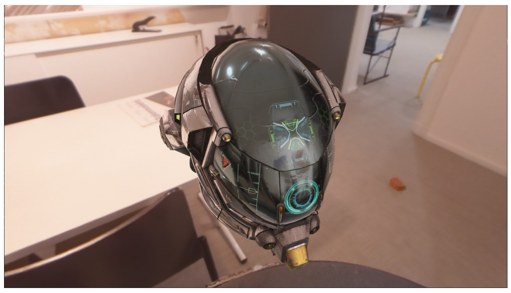

(d) Optical flow HDR using nearby warped HDRs

Figure 5.11: Nearest HDR image vs. optical flow HDR vs. SDR ground-truth(tonemapped)

As it is vital to evaluate the performance of the IBL generated by the optical flow system some results are provided for comparison. As target image the

SDR image utilized for IBL in Figure 5.11.a has been chosen and it served as background in all of the images.

In Figure 5.11.b the nearest HDR light probe was applied for IBL, leading to acceptable results, which could be improved by correcting the rotation of the cube map. This serves only as an example and the performance could improve if the light probe would be located closer to the target image, and vice versa.

The light probe interpolation system is convincing in the task of increasing the capturing density, generating additional light probes between a pair of existing ones. In Figure 5.11.c the two nearest HDRs' served as input as they were recorded, whereas in Figure 5.11.d two created warps (with Part II of the system) on a path were applied in the optical flow process. Figure 5.11.c shows roughly fitting light sources in the specular reflections, which suffer from partly blurry regions as artifacts occurred due to the differences in the adjacent HDRs. The version in Figure 5.11.d has the advantage that the rotation and perspective of them were already quite similar, leading to lower intensity of the optical flow vectors resulting in smoother images.

5.2 Quantitative results

5.2.1 Structural similarity

The structural similarity index (SSIM) is a utilized for predicting the perceived equality of any kind of digital images and videos [97]. The difference to other comparison techniques like, mean squared error (MSE) or peak signal to noise ratio (PSNR) is that these compare absolute errors between images; whereas SSIM is a perception-based model that compares perceived change in structural information, and additionally including perceptual phenomena, both luminance and contrast factors. Structural information recognizes pixels which are nearby as strongly connected, and uses these spatially close pixels as the source of information for the structure of perceived objects [96].

$$ssim\_value = ssim(A, ref) \tag{5.1}$$

Equation 5.1 computes the SSIM value for image A using ref as the reference image in Matlab. The value 1 displays the optimal result, whereas 0 symbolizes complete difference.

5.2.2 Mean squared error

The MSE between two signals X and Y with n amount of samples is typically computed as [5]:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2. \tag{5.2}$$

Equation 5.3 computes the MSE value for image A using ref as the reference image in Matlab.

$$mse\_value = immse(A, ref) \tag{5.3}$$

Lower MSE values express superior results. Zero resembles a perfect match, whereas values <30 are still acceptable.

5.2.3 Peak signal to noise ratio

PSNR is another widely used metric, which takes the maximum value of the signal into consideration, and can be described based on MSE as [5]:

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{5.4}$$

MAX_I is the maximum allowable signal intensity, divided by the square root of the MSE.

$$psnr\_value = psnr(A, ref) \tag{5.5}$$

Equation 5.5 computes the PSNR value for image A using ref as the reference image in Matlab. Higher PSNR values indicate superior results. Values $<$20 represent low performance and $\geq$ 35 acceptable ones with ideal properties above 40.

5.2.4 Table of results

To generate images fitting for quantitative results, the camera was fixed to a specific location in the scene and the model was placed in a constant position. The different metrics (SSIM, MSE, and PSNR) were all calculated on renderings of the scene, followed by tone mapping.

These renderings were generated for the calculated image, which was created with the presented solution, and for the 'real'-HDR-image recorded on the identical location as the SDR target frame. Obviously, the 'ideal'-HDR-image was not applied during the image generation and only served as input for comparison.

For these comparative renderings, the scene background was removed and only the illuminated model served as input as seen in Figure 5.12.

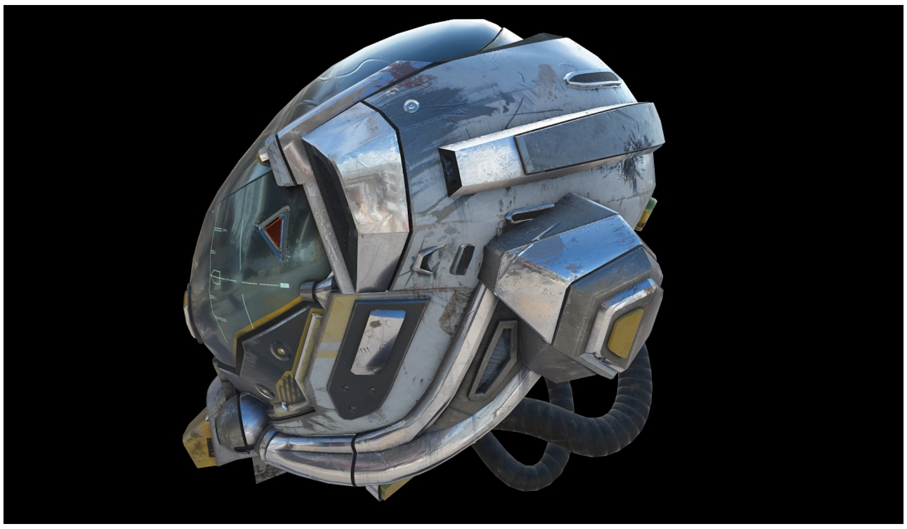


Figure 5.12: Example input for quantitative comparison

Evaluation of metrics			
Image or Metric	SSIM	MSE	PSNR
Kitchen Scene - Helmet	0.959	257.1	24.0
Kitchen Scene - Corset	0.999	2.3	44.4
Kitchen Scene - Duck	0.999	6.8	39.8
Kitchen Scene - Chair	0.974	28.1	33.6
Kitchen Scene - Shield	0.974	149.1	26.4
Flat Scene - Helmet	0.925	331.8	22.9
Flat Scene - Corset	0.996	10.5	37.9
Flat Scene - Duck	0.996	34.8	32.7
Flat Scene - Chair	0.963	39.6	32.2
Flat Scene - Shield	0.953	353.8	22.6
Outdoor Scene - Helmet	0.962	106.1	27.9
Outdoor Scene - Corset	0.998	4.5	41.6
Outdoor Scene - Duck	0.999	5.3	40.9
Outdoor Scene - Chair	0.992	12.3	37.2
Outdoor Scene - Shield	0.943	83.1	28.9

Table 5.1: Comparison of different metrics results

Independently of the scene, objects with a higher proportion of diffuse reflections lead to better results in all comparative metrics. An explanation for this is that for diffuse reflection roughly fitting details are sufficient for promising results, whereas highly specular objects reflect almost every highlight of the scene.

Overall the SSIM metric showed very high similarity with values above 92% since it compares structural information rather than absolute errors. The MSE and PSNR results are mixed, due to the fact that the shield- and helmet model result in rather high differences, while the corset and duck model display excellent results. To conclude, the results highly benefit from the fact that the generated images 'only' serve as lighting information, as the results in panorama comparisons would be worse.

Kitchen Scene:

Average metrics: SSIM: 0.981; MSE: 88.7; PSNR: 33.6

In this scene with only artificial light, the SSIM metric displayed almost perfect results. MSE and PSNR evaluated excellent results in diffuse models and although the performance on a specular object could have been better, the overall results were generally pleasing.

Flat Scene:

Average metrics: SSIM: 0.967; MSE: 154.1; PSNR: 29.7

The flat scene performed the worst of the three compared scenes. Particularly MSE and PSNR showed underwhelming results, with an average of over 150 for MSE and the lowest PSNR values close to 20. On the contrary, the SSIM was only slightly lower than the competitive locations.

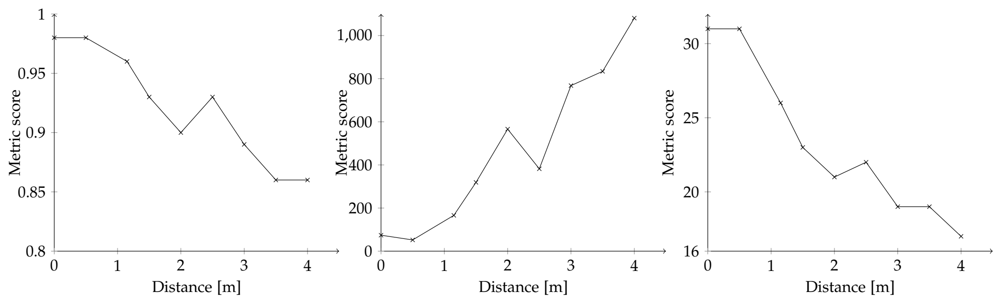
Outdoor Scene:

Average metrics: SSIM: 0.979; MSE: 42.3; PSNR: 35.3

On average, the outdoor scene outperforms the other recording locations in the evaluations that compare absolute errors between images. On all models, the rendering comparisons showed satisfying results, with possible improvements on specular objects.

5.2.5 Evaluation of image distance between HDR and SDR

It is vital to acknowledge that HDR images can only be created successfully if there is fitting HDR image information available to utilize in the algorithms. If the distance between recorded HDR images and the target frames increases, less information is obtained, as the images have fewer features in common.



(a) SSIM (b) MSE (c) PSNR

Figure 5.13: Evaluation of HDR to SDR image distance

As expected, the comparative metrics show lower performance with increasing distance. SSIM shows performance at the higher range of the scale, as the overall structural information is heavily influenced by the model's structure. In the MSE and the PSNR metric, the absolute differences are compared and so the distance has a larger impact on their results. Overall the evaluation of the metrics displays higher scores than assumed, considering partly failed warped sides after 1.5 meters distance.

The increase of performance at 2.5 meters is due to the successful creation of the top cube side, which failed at 2.0 meters as a lamp was at the ceiling in the same location. Especially if the scene is a narrow area, certain parts of the scene are not present on the available HDR image, due to the angle of the camera, or occlusions by other objects. In outdoor scenes, it may display better performance as objects are typically visible in a larger range compared to an indoor scene, with only a few meters of total area.

5.3 Memory

As previously mentioned, HDR information is a demanding task in many ways, even for modern systems. Keeping a high amount of HDR frames in the memory results in a completely unusable system during the calculation. At certain points, the available 32 gigabytes (GB) of random access memory (RAM) is fully loaded and some of the utilized HDR frames are swapped out to the solid-state drive (SSD). This outsourcing of the files leads to a slower runtime of the calculation process, due to lower read and write speed.

Each warped image can be calculated independently from each other. If the closest HDR frame is though as well the closest for another SDR frame, it can be loaded into the storage only once. Memory space may be saved by reusing the extracted feature points as they stay the same over time.

Roughly one HDR frame of the kitchen scene has 80 megabytes, which leads to about 1.4 GB of disk space for a single image, rotated in 20-degree changes. Depending on the scene between 13-17 camera positions were recorded, leading to a huge amount of data. It is important to consider, that also the resulting images are HDR frames, therefore, taking up additional memory.

These factors were the reason for downscaling in the mobile application and that HDR was only utilized for IBL. At the current state, the system is relatively unoptimized concerning memory, therefore, many options still exist. Optimization for memory structure is always a difficult task. A possible idea is, if it is known that in two images a feature point is identical, a storage structure may be created that points straight on the same memory address possibly avoiding loading times.

5.4 Runtime

There are several factors influencing the runtime of the HDR light probe generation. It is mainly affected by image resolution, HDR light probes amount, number of rotations per HDR light probe, and the algorithm choice (warping or optical flow).

Light probe interpolation The light probe interpolation process took with the resolution of 512x256 pixels 9 seconds. The optical flow calculations took most of the execution time with 5.2 seconds and the graph-cut calculated about 3.2 seconds. The advantage of this method is that there is no need for extra transformations to neither the input nor the final image as they remain as an equirectangular image. Here, the time to calculate a single side is not relevant in a significant manner, as the whole picture is calculated at once. Only the resolution of the three images may be altered, compared to the feature matching part where rotations, amount of light probes, and the resolution may be changed.

To demonstrate the scaling of the system, the execution time takes approximately 1500 seconds ($\sim$ 25 minutes) for an extremely large image of 7776x3888 pixels. In detail, about 23 seconds were spent on image loading and tone mapping, 1210 seconds on optical flow calculations, and 260 seconds for the graph-cut at this high resolution.

Feature matching and warping One complete image of feature matching and warping, considering four rotations, meaning 90-degree steps of the nearest HDR takes about 2.1 seconds using images with the resolution of 512x256 pixels. In detail, in this setup about 0.04 seconds were spent on image loading and tone mapping, 0.28 seconds on feature matching, and 0.15 seconds for image warping for a single side of the target cube map (172x172 pixels). This has to be repeated for five of the six cube sides, neglecting the bottom side leading to the mentioned $\sim$two seconds execution time. Both loading and matching times are affected by the amount of HDR light probe frames, as eight rotations roughly double the runtime. It took 0.08 seconds loading time and 0.62 seconds for feature matching while image warping took the same duration. The loading times can be excluded from the runtime, as normally they are loaded once at the beginning of the system. Compared to the light probe interpolation system the runtime is only about 25%.

The performance of the warping and matching usually declines with a low amount of input frames and the rotation process was applied once for each image and not conducted for every single execution. To rotate a 512x256 pixel panorama in four different angles it took about five seconds whereas the extreme 24 rotations took about 30 seconds.

The execution time with a higher resolution of 972x486 pixels finishes after about 30 seconds. To evaluate the scaling of this system, the runtime for a extremely high resolution image and utilizing a high amount of rotations is also presented. One iteration of feature matching and warping, considering 24 rotations, meaning 15-degree steps of the nearest HDR takes about 350 seconds ($\sim$ 6 minutes) using images with the highest available resolution of 7776x3888 pixels. In detail, 330 seconds were spent on feature matching, and 14 seconds for image warping.

Moreover, the execution time considering an additional HDR light probe would double, as the algorithm handles extra HDR frames the same way as it works with extra rotations. Therefore, considering four rotations from each of the two light sources takes the same amount of execution time as eight rotations from one light probe.

The resulting runtimes are measured considering a rather unoptimized system which does not utilize GPU acceleration and consists mainly of OpenCV functions as well as Matlab code. Related research to accelerate OpenCV calculations via GPU implementation is available and the presented execution times could be drastically improved by adding this functionality.

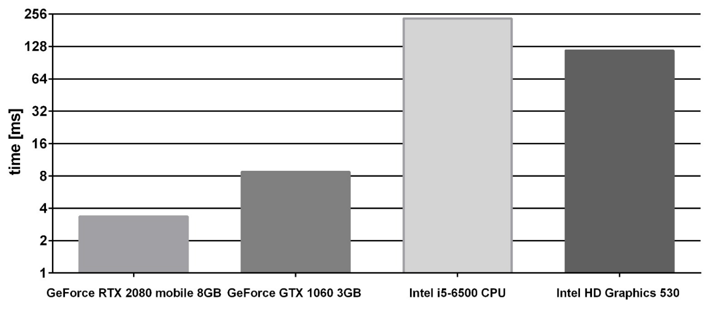


Figure 5.14: Performance results for GPU vs. CPU in the standard OpenCV core GEMM performance test modified from [11]

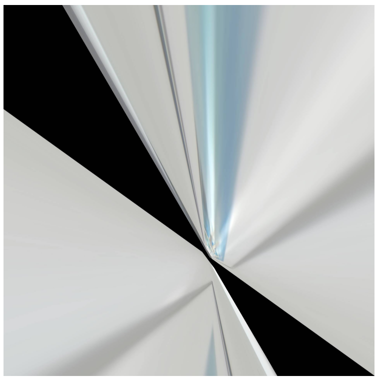
For example Bowely [11] presented results for comparison purposes, of GPU as well as CPU runtime tests (Figure 5.14). The provided graph depicts a clear acceleration of runtime in the standard OpenCV core GEMM performance test, as shorter bars represent faster execution times. To conduct this addition an NVIDIA GPU is required and several additional libraries as well as toolkits have to be built including some extra configurations. By implementing this acceleration the presented system may provide 30 fps results in certain configurations. Considering resolution, in most of the MR use cases, a rather low-resolution light probe is sufficient for credible results. For evaluation purposes, these enormous frames were also applied to display the scaling of the system.

5.5 Limitations

The previously displayed images show results in well-working scenes and adjusted parameters. However, several errors occurred during the process of the HDR generation, which are reflected in some of the evaluation metrics results. Due to a lack of features, similar objects, or non-optimally chosen parameters, failures arise in the system.

In Figure 5.15, some of the unsuccessfully created images are displayed for further analysis. Suboptimal parameter choice affects a) the number of detected features, where a lack of them leads to holes, and a surplus to mismatching, as well as b), the matching quality, causing duplicate objects or even abnormal distortions.

Mismatching is a common issue, which can be limited by utilizing AR makers and choosing feature-dense scenes. Algorithmically the resulting warp may be compared to the SDR target image to confirm that the images resemble each other, or RANSAC may be applied to filter the features. Fine-tuning of the parameters and information is key to a fitting result.


(a) Failed warping: strong distortions

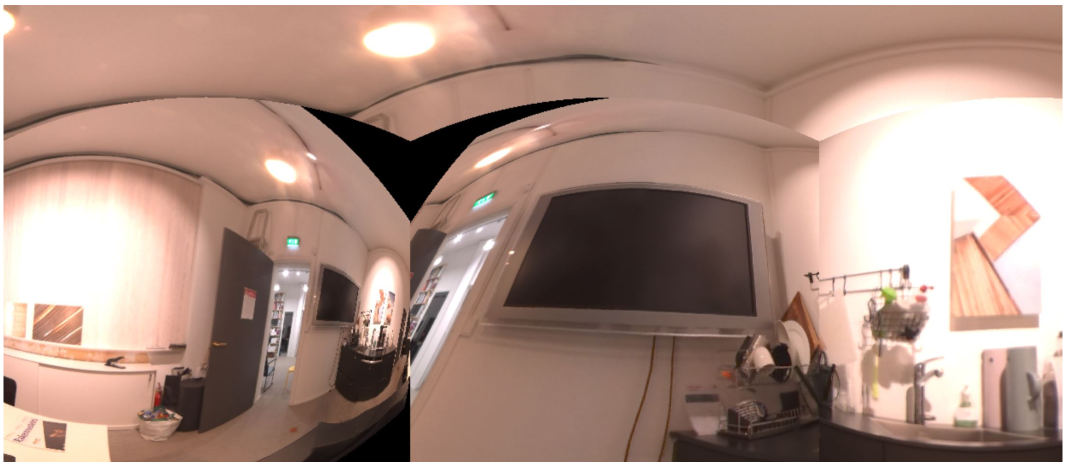

(b) Warping errors: duplicate objects, distortions and holes

Figure 5.15: System limitations

5.6 Additional comparative results

In this section the presented results are compared to existing AR applications: 1) Ikea Place application and 2) Google Hello AR Unity application.

As the model of choice, the previously presented Ikea chair *Poäng* was selected, as the model was included in the Ikea application and as well provided as a downloadable model. The *Hello AR* application was modified to display the chair model instead of the originally included one. In subsection 5.6.3 the chair model is shown in the application which was developed during this thesis, and its performance is evaluated against the existing competitors, developed by several corporations.

The resulting images were recorded in the same room and it was ensured that the 'real-world' lighting conditions and the model's position were similar. An additional chair is present in the image to display the lighting effects on a physically existing object.

5.6.1 Ikea AR application

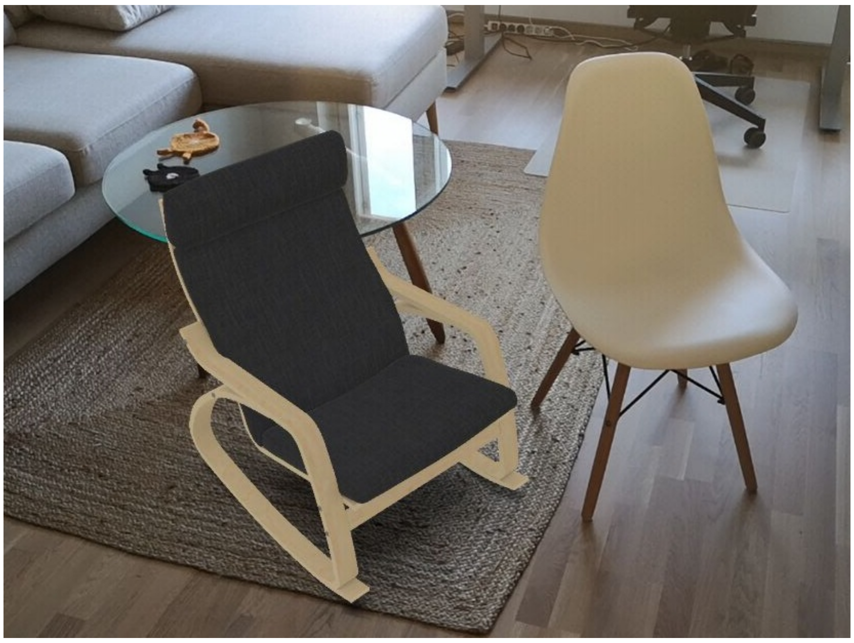


Figure 5.16: A real scene with a virtual Ikea chair Poäng. Image from Ikea Place application.

The virtual chair which was placed in the application lacks any lighting information from the real world. Thereby, the lighting is pre-defined and consists only of a static shadow below the chair and otherwise uniform illumination of the model. The chair's placement was conducted with the included ray-tracing functionality, which creates the chair at the position where the smartphone's camera is pointing at when the user touches the screen. Surprisingly, even a basic shadow has a positive impact on the perceived realism of the object. Overall, the lack of lighting and reflections leads to a distinct separation from real-world objects.

5.6.2 Google Hello AR application

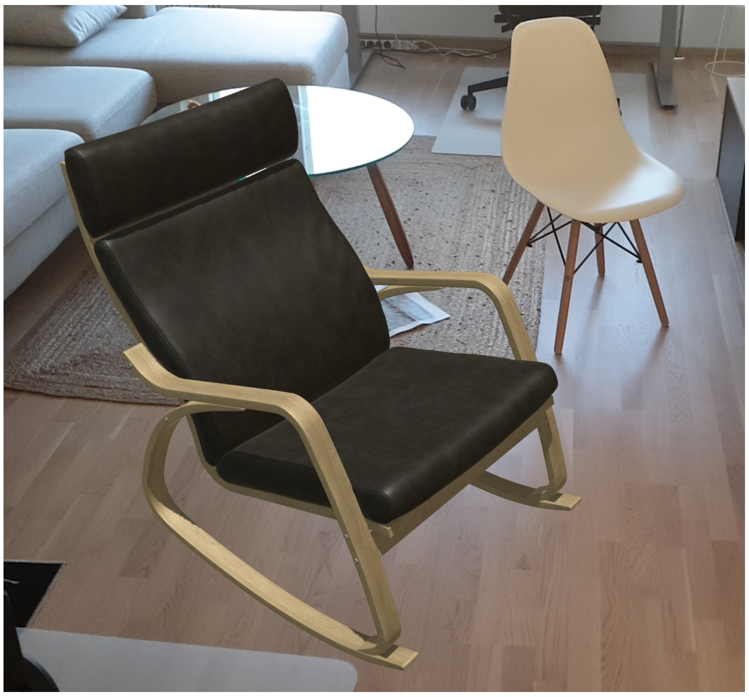


Figure 5.17: A real scene with a virtual Ikea chair Poäng. Image from ARCore application, replaced standard model with chair Poäng.

The 'Hello AR' application features a basic light estimation of the captured image, and applies it to a virtual model, which in this case was replaced with the same model as in the Ikea AR application. The application suggests flat areas as possible placement options for the virtual object, which the user can select by touching the chosen location on the screen. The model itself looks already quite real, although consistencies with the real scene could be improved, as it seems to be floating in the scene, due to the lack of shadows and impact on the real scene.

5.6.3 Own Unity AR application

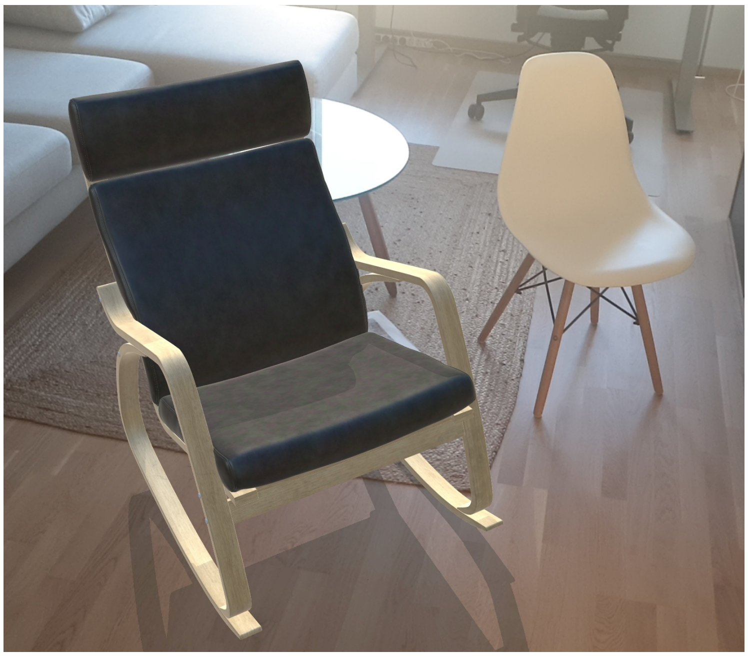


Figure 5.18: A real scene with a virtual Ikea chair Poäng. Image from created application using IBL.

Contrary to the previous applications, the presented application includes IBL from the priorly recorded HDR scene leading to an accurate representation of the scene's lighting scenario. The chair is rendered at the recognized position of a printed AR marker, which can be rotated and moved during runtime. An additional directional light source from the main lighting direction paired with a transparent plane results in realistic shadows. This technique mimics a better connection between virtual and real-world improving consistencies.

6 Conclusion and Outlook

IBL and HDR are high-performance tasks for mobile devices, especially if the result should look as lifelike as possible. In accordance with Moore's law, which predicted exponential growth of transistor count in integrated circuits, also overall development in information technology has shown similar advances. The performance of mobile devices is improved at a relatively quick rate, but if the device capacities are limited modern streaming technologies might support in non-real-time fields of application [80].

When aiming for IBL, the generation of HDR images with the presented method can be recommended, even if the method may produce some more apparent artifacts (versus SDR images), the results outperform the standard illumination. Artifacts that might be visibly noticeable in the resulting images are most of the time of minor impact in the model's lighting.

Likewise noticeable were the variances in results obtained during this practical project. The results were varying quite a lot depending on the scene. Dynamic scenes cause lots of artifacts if the camera is moving fast, and objects differ greatly. Therefore, this aspect needs to be further analyzed to reduce those artifacts.

6.1 Benefits

The implementation of the results in this thesis are highly beneficial in two potential areas, namely *computer-generated imagery movie illumination* and MR.

In the movie industry, especially lighting techniques have a big impact on the resemblance to real-world scenes. There are some expensive hardware solutions available to deal with this topic, but an extremely cheaper version using algorithms instead of expensive and inconvenient hardware could be an acceptable alternative. The price of the *Insta360 Titan*, which can record HDR videos is currently 16000 euros whereas the Samsung Gear 360° camera, which was utilized in this thesis costs roughly 200 euros.

In AR, MR, and as well VR, more realistic illumination would be a benefit for virtual scenes and generated objects. It could be used for example with the *Microsoft Hololens* [58], to apply coherent and lifelike illumination to virtual objects in the room. As seen in the comparison to the Ikea app, which uses only basic lighting on their virtual furniture, the results of this project highlight the big impact of IBL.

6.2 Future Work

Quality of life features Currently, the framework relies on input images by the user which have to be extracted, separated, and converted by using different software like *Shotcut* (section 4.5). Therefore it might be a useful feature to extend the framework by an automatic routine to open the different programs and run them with the necessary files.

Another quite inconvenient step concerning the creation of the HDR images is its dependency on manual exposure fusion, as Picturenaut (section 4.5) needs given exposure values per image to correctly fuse them. This requires additional time during the setup of the system as these steps have to be repeated for every HDR frame. This part could be improved by automation, requiring a set sequence of images with no 'mistakes' during recording, to assign the frame sequence always to the same fitting sequence of exposure values. Alternatively, this step can be simplified by the usage of an HDR-compatible 360° camera, which is already existing but has inconvenient features e.g. high cost, large size, and impracticability for an average consumer.

A possible quality of life feature is a graphical user interface or a fully deployable program, as the end consumer might not have the knowledge or necessary licenses to run *Matlab* or other required software.

Improvements Consistencies within frame sequences are currently not taken into account, as each of the pictures is created individually. Further research in this field could lead to an even more realistic result, as minor interruptions in the picture sequence could disrupt the flow of the object's illumination. Currently, each frame is created by itself, but the usage of further information given by adjacent or even the whole sequence could reduce possible artifacts, like flickering caused by drastic changes from one frame to the other. In this part, some threshold in the difference between frames could have been implemented to discard images that have low similarity with their warped neighbors. If the difference is too high, the frame in question is replaced by a frame that is created with the optical flow algorithm instead.

During the practical process of this thesis, Colmap did not yet support equirectangular input images for scene reconstruction, therefore only the average position per image is applied, which might not be as accurate as a localized 360° image. This could further increase the robustness and performance.

Real-world scenes tend to change over time, especially outside the lighting conditions vary drastically. Therefore, some mechanism to update the existing light probes and update the whole data structure would be required, especially if lightning changes are detected. The sampling of the scene would require extra research, to find out the amount with which the performance stays in an acceptable range. Every redundant HDR capturing avoided, reduces effort in the initial setup phase. Not only the sampling amount but also well the optimal sampling locations might help to improve these issues. Potential viewing angles and the structure of the scene might offer helpful insight during recording.

Extension with VR As mentioned in subsection 2.1.3 the main difference between AR and VR is that AR requires only a camera, while VR needs a headset for displaying. Therefore, an extension for VR requires an adaptation that also facilitates the display of the scene on the device, instead of only the virtual model. The localization in the scene would in this case be unnecessary, as no input from the real-world scene exists.

An aspect to consider while using VR is that the displayed information should be coherent with everyday situations, which means smooth scenes, as few video jitters as possible, and no sudden unnatural jumps, as this may cause nausea.

All these aforementioned features have to be taken into consideration for further implementations of the presented framework. These results may therefore have a high impact on further research in this field of application.

Bibliography

[1] JetRuby Agency. *Augmented Reality vs Virtual Reality: A Short Comparison*. URL: https://expertise.jetruby.com/augmented-reality-vs-virtual-reality-comparison-b4b0cc923d9f (cit. on p. 11).

[2] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. "FREAK: Fast retina keypoint." In: June 2012, pp. 510–517. DOI: 10.1109/CVPR.2012.6247715 (cit. on p. 27).

[3] Abid K. Revision Alexander Mordvintsev. *Feature Matching with OpenCV*. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html (cit. on pp. 42, 54).

[4] Shai Bagon. *Matlab Wrapper for Graph Cut*. https://github.com/shaibagon/GCMex. December 2006 (cit. on pp. 48, 52).

[5] Francesco Banterle et al. *Advanced High Dynamic Range Imaging: Theory and Practice*. 1st. Natick, MA, USA: A. K. Peters, Ltd., 2011. ISBN: 9781568817194 (cit. on pp. 51, 70).

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded up robust features." In: vol. 3951. July 2006, pp. 404–417. DOI: 10.1007/11744023_32 (cit. on p. 27).

[7] Van De Sluis Berent Willem Meerbeek; Bartel Marinus. "Image - Based Lighting." In: (Sept. 2019) (cit. on p. 24).

[8] Pravin Bhat et al. "Using Photographs to Enhance Videos of a Static Scene." In: *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*. Ed. by Jan Kautz and Sumanta Pattanaik. Eurographics. Grenoble, France, June 2007, pp. 327–338. URL: http://www.cs.washington.edu/homes/pro/papers/videoEnhancement/videoEnhancement.html (cit. on p. 47).

[9] Ma Yuan Bian Yu. "Dynamic daylighting simulation facility based on image-data." In: *Journal of Tsinghua University(Science and Technology)* 60.7, 597 (2020), p. 597. DOI: 10.16511/j.cnki.qhdxxb.2019.26.038. URL: http://jst.tsinghuajournals.com/EN/abstract/article_153620.shtml (cit. on p. 25).

[10] Atanas Boev. *Augmented Reality*. URL: http://www.cs.tut.fi/kurssit/SGN-5406/lectures/VR9-augmented-reality.pdf (cit. on p. 46).

[11] James Bowely. *Accelerate OpenCV 4.4.0 build with CUDA and python bindings*. URL: https://jamesbowley.co.uk/accelerate-opencv-4-4-0-build-with-cuda-and-python-bindings/ (cit. on pp. 77, 78).

[12] Yuri Boykov and Vladimir Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision." In: *In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 26, no. 9, September 2004, pp. 1124-1137*. IEEE. 2004 (cit. on p. 52).

[13] Cghugge. *IKEA Poang rocking chair*. URL: https://sketchfab.com/3d-models/ikea-poang-rocking-chair-f74c8e97e8514d8fac6b4860272b1a76 (cit. on pp. 59–62).

[14] Sagar Rohidas Chavan. "Augmented Reality vs. Virtual Reality: Differences and Similarities." In: 2016 (cit. on p. 12).

[15] Choyg. https://github.com/whdlgp/Equirectangular_rotate (cit. on p. 54).

[16] VRM Consortium. *VRM*. URL: https://vrm.dev/en/docs/univrm/install/univrm_install/ (cit. on p. 59).

[17] R. L. Cook and K. E. Torrance. "A Reflectance Model for Computer Graphics." In: *ACM Trans. Graph.* 1.1 (Jan. 1982), pp. 7–24. ISSN: 0730-0301. DOI: 10.1145/357290.357293. URL: https://doi.org/10.1145/357290.357293 (cit. on p. 14).

[18] BenQ Corporation. *BenQ EX3501R*. URL: https://www.benq.com/en-us/monitor/gaming/ex3501r.html (cit. on p. 2).

[19] Chen Cui. "MRI fat-water separation using graph search based methods." In: 2017 (cit. on p. 30).

[20] P. Debevec and Jitendra Malik. “Recovering high dynamic range radiance maps from photographs.” In: *SIGGRAPH ’97*. 1997 (cit. on pp. 21, 22).

[21] Paul Debevec. “Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography.” In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 189–198. ISBN: 0897919998. DOI: `10.1145/280814.280864`. URL: `https://doi.org/10.1145/280814.280864` (cit. on p. 23).

[22] *Detection of optic flow in the image plane*. `http://www.scholarpedia.org/article/File:FigIntroOpticFlow.png`. Accessed: 2013-05-16 (cit. on p. 18).

[23] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description.” In: (Dec. 2017) (cit. on p. 27).

[24] G. Eilertsen, R. K. Mantiuk, and J. Unger. “A Comparative Review of Tone-mapping Algorithms for High Dynamic Range Video.” In: *Comput. Graph. Forum* 36.2 (May 2017), pp. 565–592. ISSN: 0167-7055. DOI: `10.1111/cgf.13148`. URL: `https://doi.org/10.1111/cgf.13148` (cit. on p. 21).

[25] Selim Esedoglu and Jianhong Shen. “Digital inpainting based on the Mumford-Shah-Euler image model.” In: *EUROPEAN J. APPL. MATH* 13 (2002), pp. 353–370 (cit. on p. 54).

[26] Facebook. *Instagram*. URL: `https://www.instagram.com` (cit. on p. 1).

[27] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew Davison. “KAZE Features.” In: Oct. 2012. DOI: `10.1007/978-3-642-33783-3_16` (cit. on p. 27).

[28] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.” In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: `10.1145/358669.358692`. URL: `https://doi.org/10.1145/358669.358692` (cit. on p. 48).

[29] Twentieth Century Fox. *Alita: Battle Angel*. URL: `https://www.vulture.com/2018/12/alita-battle-angels-cgi-eyes-behind-the-design.html` (cit. on p. 2).

[30] Google. *ARCore*. URL: `https://developers.google.com/ar/discover` (cit. on p. 55).

[31] Google. *Filament*. URL: `https://google.github.io/filament/Filament.md.html` (cit. on p. 12).

[32] Ringo Gunther. *6th - Reflection // Ikana's Mirror Shield*. URL: `https://sketchfab.com/3d-models/6th-reflection-ikanas-mirror-shield-8ac5f008199e4b888649501921d3a82b` (cit. on pp. 59–62).

[33] Ankit Gupta et al. "Enhancing and Experiencing Spacetime Resolution with Videos and Stills." In: *International Conference on Computational Photography*. IEEE. San Francisco, France, 2009. URL: `http://grail.cs.washington.edu/projects/enhancing-spacetime/` (cit. on pp. 19, 33, 37, 47).

[34] Alex Hart. *Image-based lighting on CG characters*. URL: `http://eahart.com/portfolio/image-based-lighting-on-cg-characters` (cit. on p. 2).

[35] M. Hassaballah, Abdelmgeid Ali, and Hammam Alshazly. "Image Features Detection, Description and Matching." In: vol. 630. Feb. 2016, pp. 11–45. ISBN: ISBN 978-3-319-28852-9. DOI: `10.1007/978-3-319-28854-3_2` (cit. on pp. 16, 17).

[36] hdrlabs.com. *Picturenaut*. URL: `http://www.hdrlabs.com/picturenaut/` (cit. on pp. 4, 55).

[37] Jerry Hildenbrand. `https://www.androidcentral.com/what-hdr10-and-why-does-it-make-galaxy-s10-screen-better`. August 2019 (cit. on p. 2).

[38] Hristina Hristova et al. "Color transfer between high-dynamic-range images." In: (Aug. 2015). DOI: `10.1117/12.2186774` (cit. on p. 28).

[39] Yanzhu Hu et al. "An Improved Multithreshold Segmentation Algorithm Based on Graph Cuts Applicable for Irregular Image." In: *Mathematical Problems in Engineering* 2019 (May 2019), pp. 1–25. DOI: `10.1155/2019/3514258` (cit. on p. 31).

[40] John F. Hughes et al. *Computer Graphics: Principles and Practice*. 3rd ed. Addison-Wesley, 2013. ISBN: 978-0-321-39952-6 (cit. on pp. 13, 16, 45).

[41] Anaconda Inc. *Anaconda*. URL: `https://anaconda.org/` (cit. on p. 48).

[42] Mathworks Inc. *Matlab*. URL: `https://www.mathworks.com` (cit. on pp. 19, 47, 51).

[43] Snap Inc. *Snapchat*. URL: `https://www.snapchat.com/l/en-gb/` (cit. on p. 1).

[44] Zeeshan Khan, Mukul Khanna, and Shanmuganathan Raman. "FHDR: HDR Image Reconstruction from a Single LDR Image using Feedback Network." In: *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2019), pp. 1–5 (cit. on p. 23).

[45] KhronosGroup. *glTF 2.0 Sample Models*. URL: `https://github.com/KhronosGroup/glTF-Sample-Models/tree/master/2.0` (cit. on pp. 45, 59–63).

[46] Tom Kneiphof, Tim Golla, and Reinhard Klein. "Real-time Image-based Lighting of Microfacet BRDFs with Varying Iridescence." In: *Computer Graphics Forum* 38 (July 2019). DOI: `10.1111/cgf.13772` (cit. on p. 25).

[47] Vladimir Kolmogorov and Ramin Zabih. "What Energy Functions can be Minimized via Graph Cuts?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 26, no. 2, February 2004, pp. 147-159*. IEEE. 2004 (cit. on p. 52).

[48] Vivek Kwatra et al. "Graphcut Textures: Image and Video Synthesis Using Graph Cuts." In: *ACM Transactions on Graphics, SIGGRAPH 2003* 22.3 (July 2003), pp. 277–286 (cit. on p. 20).

[49] Siyeong Lee, Gwon Hwan An, and Suk-Ju Kang. "Deep Chain HDRI: Reconstructing a High Dynamic Range Image from a Single Low Dynamic Range Image." In: *IEEE Access* 6 (2018), pp. 49913–49924 (cit. on p. 23).

[50] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. "BRISK: Binary Robust invariant scalable keypoints." In: Nov. 2011, pp. 2548–2555. DOI: `10.1109/ICCV.2011.6126542` (cit. on p. 27).

[51] Chenge Lexi Li et al. “TrackNet: Simultaneous Object Detection and Tracking and Its Application in Traffic Video Analysis.” In: (Feb. 2019) (cit. on p. 30).

[52] C. Liu. “ Beyond Pixels: Exploring New Representations and Applications for Motion Analysis.” In: *Doctoral Thesis. Massachusetts Institute of Technology.)* 2009 (cit. on pp. 19, 48, 51).

[53] Pengpeng Liu et al. “SelFlow: Self-Supervised Learning of Optical Flow.” In: *CVPR*. 2019 (cit. on p. 29).

[54] D. G. Lowe. “Object recognition from local scale-invariant features.” In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2 (cit. on p. 27).

[55] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints.” In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94 (cit. on p. 42).

[56] *Luminance HDR*. https://luminancehdr.readthedocs.io/en/latest/. Accessed: 2017-05-03 (cit. on pp. 4, 56).

[57] Shay Maymon and Hila Barel. “Contrast Optimization And Local Adaptation (COALA) for HDR Compression.” In: *ArXiv* abs/1905.06372 (2019) (cit. on p. 23).

[58] Microsoft. *Hololens*. URL: https://docs.microsoft.com/en-us/hololens/ (cit. on p. 84).

[59] Microsoft. *Visual Studio*. URL: https://visualstudio.microsoft.com/ (cit. on p. 51).

[60] Paul Milgram and Fumio Kishino. “A Taxonomy of Mixed Reality Visual Displays.” In: *IEICE Trans. Information Systems* vol. E77-D, no. 12 (Dec. 1994), pp. 1321–1329 (cit. on pp. 11, 12).

[61] S. Miller, MAGI Synthavision, and C. R. Hoffman. “Illumination and Reflection Maps : Simulated Objects in Simulated and Real Environments Gene.” In: 1984 (cit. on p. 23).

[62] Alvaro Montero et al. “Designing and Implementing Interactive and Realistic Augmented Reality Experiences.” In: *Univers. Access Inf. Soc.* 18.1 (Mar. 2019), pp. 49–61. ISSN: 1615-5289. DOI: 10.1007/s10209-017-0584-2. URL: https://doi.org/10.1007/s10209-017-0584-2 (cit. on p. 2).

[63] Anca Morar, Florica Moldoveanu, and E. Groller. “Image segmentation based on active contours without edges.” In: *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing* (2012), pp. 213–220 (cit. on p. 30).

[64] Inc. Niantic. *Pokemon Go*. URL: https://pokemongolive.com/en/ (cit. on p. 1).

[65] Peter O’Donovan. “Optical Flow : Techniques and Applications.” In: 2005 (cit. on p. 30).

[66] Yuki Ono et al. “LF-Net: Learning Local Features from Images.” In: May 2018 (cit. on p. 27).

[67] Michael Oren and Shree K. Nayar. “Generalization of Lambert’s Reflectance Model.” In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’94. New York, NY, USA: Association for Computing Machinery, 1994, pp. 239–246. ISBN: 0897916670. DOI: 10.1145/192161.192213. URL: https://doi.org/10.1145/192161.192213 (cit. on p. 14).

[68] Simone Parisotto and Carola Schönlieb. *MATLAB/Python Codes for the Image Inpainting Problem*. GitHub repository, MATLAB Central File Exchange, https://github.com/simoneparisotto/MATLAB-Python-inpainting-codes. Sept. 2016 (cit. on p. 54).

[69] Jinwoo Park et al. “Physically-inspired Deep Light Estimation from a Homogeneous-Material Object for Mixed Reality Lighting.” In: *IEEE Transactions on Visualization and Computer Graphics* PP (Feb. 2020), pp. 1–1. DOI: 10.1109/TVCG.2020.2973050 (cit. on pp. 26, 27).

[70] Raymond Phan. https://github.com/rayryeng/equi2cubic. 2015 (cit. on p. 52).

[71] Raymond Phan. https://github.com/rayryeng/cubic2equi. 2014 (cit. on p. 52).

[72] Bui Tuong Phong. "Illumination for Computer Generated Pictures." In: *Commun. ACM* 18.6 (June 1975), pp. 311–317. ISSN: 0001-0782. DOI: 10.1145/360825.360839. URL: https://doi.org/10.1145/360825.360839 (cit. on p. 14).

[73] OSC Source Creation Project. *Python and OpenCV tutorial.* URL: https://www.mdeditor.tw/pl/p38J (cit. on p. 44).

[74] E. Reinhard et al. "High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting." In: 2010 (cit. on p. 13).

[75] Erik Reinhard et al. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics).* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN: 0125852630 (cit. on pp. 10, 21).

[76] Ethan Rublee et al. "ORB: an efficient alternative to SIFT or SURF." In: Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544 (cit. on p. 27).

[77] Ehab Salahat and Murad Qasaimeh. "Recent Advances in Features Extraction and Description Algorithms: A Comprehensive Survey." In: *CoRR* abs/1703.06376 (2017). arXiv: 1703.06376. URL: http://arxiv.org/abs/1703.06376 (cit. on p. 27).

[78] Samsung. https://img.global.news.samsung.com/global/wp-content/uploads/2016/05/Gear360_Tutorial_Main_1_1.jpg (cit. on p. 35).

[79] Samsung. https://images.samsung.com/is/image/samsung/in-galaxy-s9-sm-g960-sm-g960fzkdins-frontblack-121040348?$PD_GALLERY_L_SHOP_JPG$ (cit. on p. 35).

[80] Hollerer Tobias Schmalstieg Dieter. *Augmented Reality: Principles and Practice (Usability).* Pearson Education, 2016 (cit. on pp. 11, 13, 83).

[81] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited." In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2016 (cit. on p. 56).

[82] Johannes Lutz Schönberger et al. "Pixelwise View Selection for Unstructured Multi-View Stereo." In: *European Conference on Computer Vision (ECCV).* 2016 (cit. on p. 56).

[83] *Shotcut*. https://shotcut.org/. Accessed: 2019-05-27 (cit. on p. 55).

[84] Noah Snavely. *NASA Mars Rover imageswith SIFT feature matches*. URL: https://homes.cs.washington.edu/~shapiro/EE596/notes/6_Descriptors-1-19.pdf (cit. on p. 29).

[85] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010 (cit. on pp. 8, 18, 30).

[86] Mooii Tech. *Photoscape X*. URL: http://x.photoscape.org/ (cit. on p. 56).

[87] K. E. Torrance and E. M. Sparrow. "Theory for Off-Specular Reflection from Roughened Surfaces." In: *Radiometry*. USA: Jones and Bartlett Publishers, Inc., 1992, pp. 32–41. ISBN: 0867202947 (cit. on p. 15).

[88] Prune Truong, Stefanos Apostolopoulos, and Sandro De Zanet. "Comparison of Feature Detectors for Retinal Image Alignment." In: 2019. DOI: 10.1109/ICCV.2019.01083 (cit. on p. 27).

[89] Prune Truong et al. "GLAMpoints: Greedily Learned Accurate Match Points." In: Oct. 2019, pp. 10731–10740. DOI: 10.1109/ICCV.2019.01083 (cit. on pp. 17, 28).

[90] Deepanshu Tyagi. *Introduction To Feature Detection And Matching*. URL: https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d (cit. on p. 17).

[91] Jonas Unger, Stefan Gustavson, and Anders Ynnerman. "Spatially varying image based lighting by light probe sequences: Capture, processing and rendering." In: *The Visual Computer* 23 (May 2007), pp. 453–465. DOI: 10.1007/s00371-007-0127-6 (cit. on p. 26).

[92] Joey de Vries. *Diffuse irradiance*. URL: https://learnopengl.com/PBR/IBL/Diffuse-irradiance (cit. on pp. 14, 15).

[93] Vuforia.com. *Vuforia.com*. URL: https://library.vuforia.com/getting-started/overview.html (cit. on p. 36).

[94] Daniel Wagner. *AR Marker tracking pipeline*. URL: https://www.slideshare.net/MysticMonkey/augmented-reality-beyond-the-hype/25-MARKER_TRACKING_PIPELINE_Daniel_Wagner (cit. on p. 46).

[95] Yang Wang et al. "UnOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 29).

[96] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. "Multi-Scale Structural Similarity for Image Quality Assessment." In: *in Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers, (Asilomar*. 2003, pp. 1398–1402 (cit. on p. 69).

[97] Zhou Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity." In: *Trans. Img. Proc.* 13.4 (Apr. 2004), pp. 600–612. ISSN: 1057-7149. DOI: 10.1109/TIP.2003.819861. URL: http://dx.doi.org/10.1109/TIP.2003.819861 (cit. on p. 69).

[98] Thomas Weibel et al. "Endoscopic bladder image registration using sparse graph cuts." In: *2010 IEEE International Conference on Image Processing* (2010), pp. 157–160 (cit. on p. 30).

[99] www.dpreview.com. http://www.dpreview.com/files/p/articles/3998514868/3.jpeg (cit. on p. 9).

[100] www.ptgui.com. https://www.ptgui.com/hdrtutorial.html (cit. on pp. 9, 35).

[101] Qingsen Yan et al. "Attention-Guided Network for Ghost-Free High Dynamic Range Imaging." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 22).

[102] Kwang Yi et al. "LIFT: Learned Invariant Feature Transform." In: vol. 9910. Oct. 2016, pp. 467–483. ISBN: 978-3-319-46465-7. DOI: 10.1007/978-3-319-46466-4_28 (cit. on p. 27).

[103] Ramin Zabih Yuri Boykov Olga Veksler. "Efficient Approximate Energy Minimization via Graph Cuts." In: *IEEE transactions on PAMI, vol. 20, no. 12, p. 1222-1239, November 2001*. IEEE. 2001 (cit. on pp. 48, 52).

[104] Zadrakos. *Half-Life:Alyx*. URL: https://www.youtube.com/watch?v=17dKkrt4NUU (cit. on p. 1).

[105] Yiqin Zhao and Tian Guo. "PointAR: Efficient Lighting Estimation for Mobile Augmented Reality." In: (Mar. 2020) (cit. on p. 26).

[106] Stefanie Zollmann. *COSC342:Computer Graphics University of Otago*. URL: `http://www.cs.otago.ac.nz/cosc342/2017-notes/342-2017lect15.pdf` (cit. on p. 16).