Depth Image-Based Rendering for Full Parallax Displays

Rendering, Compression, and Interpolation of Content for Autostereoscopic Poster and Video Displays

Dipl.-Ing. Daniel Jung

Dissertation
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel
eingereicht im Jahr 2014

Kiel Computer Science Series (KCSS) 2015/1 v1.0 dated 2015-01-10

ISSN 2193-6781 (print version)
ISSN 2194-6639 (electronic version)

Electronic version, updates, errata available via https://www.informatik.uni-kiel.de/kcss

The author can be contacted via http://www.mip.informatik.uni-kiel.de

Published by the Department of Computer Science, Kiel University

Multimedia Information Processing

Please cite as:

▷ Daniel Jung. *Depth Image-Based Rendering for Full Parallax Displays* Number 2015/1 in Kiel Computer Science Series. Department of Computer Science, 2015. Dissertation, Faculty of Engineering, Kiel University.

```
@book{JungD15DIBR,
  author    = {Daniel Jung},
  title     = {Depth Image-Based Rendering for
               Full Parallax Displays},
  publisher = {Department of Computer Science, CAU Kiel},
  year      = {2015},
  number    = {2015/1},
  series    = {Kiel Computer Science Series},
  note      = {Dissertation, Faculty of Engineering,
               Kiel University.}
}
```

About this Series

The Kiel Computer Science Series (KCSS) covers dissertations, habilitation theses, lecture notes, textbooks, surveys, collections, handbooks, etc. written at the Department of Computer Science at Kiel University. It was initiated in 2011 to support authors in the dissemination of their work in electronic and printed form, without restricting their rights to their work. The series provides a unified appearance and aims at high-quality typography. The KCSS is an open access series; all series titles are electronically available free of charge at the department's website. In addition, authors are encouraged to make printed copies available at a reasonable price, typically with a print-on-demand service.

Please visit `http://www.informatik.uni-kiel.de/kcss` for more information, for instructions how to publish in the KCSS, and for access to all existing publications.

1. Gutachter:	Prof. Dr.-Ing. Reinhard Koch Christian-Albrechts-Universität zu Kiel
2. Gutachter:	Prof. Dr. Andreas Kolb Universität Siegen

Datum der mündlichen Prüfung: 7. November 2014

Acknowledgements

Writing this thesis has been a long lasting and demanding effort, not only for me but also for those supporting me during times of hardship and high workload. Aside from that, the work on the thesis has been an interesting and fulfilling privilege that granted insights, only achievable through the opportunity to work a long time in one field of research.

Therefore, I would particularly like to thank my supervisor Reinhard Koch for entrusting me with this challenging and interesting topic. His advice and experience proved invaluable in many discussions about scientific problems as well as coping with the occasionally troubles of day-to-day work.

My former colleague Bogumil Bartczak introduced me to my research topic and supported me greatly in the early years, together with Anatol Frick who was always open for the discussion of difficult problems. Later I shared the office with Falko Kellner and Andreas Jordt, afterwards. The time spend together led to a fertile exchange of ideas in research topics and software development techniques which I am going to miss.

I would also like to thank my colleagues Sandro Esquivel, Johannes Brünger, Oliver Fleischmann, Stefan Reinhold, Dominik Wolters, and Claudius Zelenka as well as former colleagues Anne Jordt, Ingo Schiller, Arne Petersen, Kevin Köser, Robert Wulff, Kristine Bauer, Christian Beder, Markus Franke, Lilian Zhang, and Dan Chen who were always helpful, be it discussing an area of their expertise or fixing our software.

Our system administrator, Torge Storm, provided an excellent IT infrastructure and never failed to craft amazingly professional looking gimmicks in no time, when desperately needed. My special thanks to him for always conjuring up a place to dump some terabyte of data.

Renate Staecker did an excellent job in reducing my administrative work to a bare minimum and knew always whom to approach in order to resolve issues efficiently.

In the late phase of my thesis I would like to thank Andreas Kolb and Reinhard Koch for reviewing my thesis and their participation in the assessment commission. Their helpful comments greatly improved this thesis

and thanks to their careful review countless slip-ups and inaccuracies were eliminated. My sincere thanks is also owned to Manfred Schimmler and Reinhard von Hanxleden for their work in the assessment commission.

Last but not least I would like to thank my parents for supporting my academic choices and for always believing in me, my sisters for their counseling when needed, and of course my wife for enduring overtime hours, supporting me during times of high workload, and for just being there when needed. My special thanks to my beloved ones, Julius Alexander and Viktoria Melissa Persephone, for reminding me of what is important in life.

Abstract

Advancements in production and display techniques allowed for novel displays to emerge that project a high-resolution light field for static poster content and video content, as well. These displays allow a full parallax, hence an audience can perceive a stereoscopic view of a scene without special glasses, which adjusts to the observer's position. The application of such displays are public places where the audience does not wear special glasses and is not restricted in movement.

The rendering, storage, and transfer of the large amount of data required by those displays is a challenge. The image data for a static poster display is about 200 GB and the data rate for video displays are to be expected two to four orders of magnitude higher than HDTV.

In this work the challenges are met by utilising Depth Image-Based Rendering to reduce the amount of data at the very beginning, during rendering. A fraction of the full amount of colour and depth images are rendered and used to interpolate the full data set. The rendering with state of the art ray tracers is described and a novel method to render image data for full parallax displays using OpenGL is contributed, that addresses some shortcomings of previous approaches.

For static poster displays a scene based representation for image interpolation is introduced, which efficiently utilises multi-core processors and graphics hardware for parallelization, found on modern workstations. The introduced approach implements lossy compression of the input data, and handles arbitrary scenes, using a novel Best-Next-View selection algorithm.

For video displays the real-time constraint does not allow for a costly interpolation or scene analysis. Hence, a novel approach is presented that uses a basic and computational inexpensive interpolation, and combines the interpolation results of different image representations without introducing prominent artefacts.

Zusammenfassung

Verbesserte Produktionstechniken und hochauflösende Anzeigegeräte ermöglichen Poster- und Videoanzeigen die ein hochauflösendes Lichtfeld projizieren. Dieses ermöglicht einer Vielzahl von Personen das stereoskopische Betrachten einer Szene, die sich an die Position des Betrachters anpasst, ohne das spezielle Sehhilfen benötigt werden.

Die benötigten Bilddaten einer solchen Anzeige sind enorm und stellen hohe Anforderungen an die Bilderzeugung, den Speicherbedarf und die zur Übertragung benötigte Bandbreite. Eine Posteranzeige benötigt circa 200 GB an Bilddaten und die Datenrate zukünftiger Videoanzeigen wird auf das Hundert- bis Zehntausendfache von HDTV prognostiziert.

Diese Arbeit untersucht die Möglichkeit tiefenkompensierte Interpolation zu verwenden, um die Daten schon bei der Bilderzeugung zu reduzieren. Der volle Datensatz wird dabei aus einem Bruchteil der Farb- und Tiefendaten interpoliert. Die Bilderzeugung mittels Ray Tracer wird erläutert, sowie eine neuartige Methode zur Bilderzeugung mittels OpenGL, die Nachteile bisher beschriebener Verfahren vermeidet.

Zur Bildinterpolation für Posteranzeigen wird eine szenenbasiert Darstellung entwickelt, die die Mehrprozessorsysteme und Grafikhardware moderner Arbeitscomputer effizient nutzt. Der Interpolationsalgorithmus ermöglicht ferner eine verlustbehaftete Komprimierung der Eingabedaten und, aufgrund einer Szenenanalyse, die Verwendung beliebiger Szenen.

Videoanzeigen benötigen eine Bildinterpolation in Echtzeit, wodurch aufwendige Interpolationsverfahren oder eine Szenenanalyse nicht verwendet werden können. Daher wurde eine neuartige Methode entwickelt, die eine einfache Interpolation verwendet und durch die Kombination der Interpolationsergebnisse verschiedener Bildrepräsentationen auffällige Bildartefakte vermeidet.

Contents

Acronyms

AD	Average Distance	36, 68, 76, 77, 79, 81, 82, 90, 91, 93, 94, 128, 137, 138, 151, 155
AOF	Angle Of Field	32, 145, 211
API	Application Programming Interface	24, 26, 38, 42
ASIC	Application-Specific Integrated Circuit	50
BNV	Best-Next-View	3, 5, 113, 115–118, 122, 126–133, 164, 165, 212, 213
CCD	Charge-Coupled Device	20
Cg	C for Graphics	26
CPU	Central Processing Unit	4, 38, 39, 101, 111, 112, 159, 163, 164, 183, 185
CUDA	Compute Unified Device Architecture	26
DIBR	Depth Image-Based Rendering	2–5, 42, 46, 58, 64, 83, 99, 100, 112, 115, 117, 118, 120, 122, 123, 125, 126, 129, 132, 133, 135, 136, 144, 163–165
DPI	Dots Per Inch	11, 12
DVI	Digital Visual Interface	161, 165
FCP	Far Clipping Plane	28

FOV	Field Of View	44, 68, 87, 88, 90, 95, 97, 103, 105, 119, 128, 142, 143, 171, 172, 177, 211, *Glossary:* ψ
FPGA	Field-Programmable Gate Array	5, 38, 50, 147, 148, 159, 160, 164, 165
FPS	Frames Per Second	2, 48, 49, 159, 161, 213
GLSL	OpenGL Shading Language	26
GPGPU	General-Purpose computing on Graphics Processing Unit	5, 24–26
GPU	Graphics Processing Unit	4–6, 24, 26, 38, 39, 50, 101, 111, 112, 163, 164
HDTV	High Definition Television	vii, ix, 49, 161
IBR	Image Based Rendering	42, 43, 45, 46, 49, 64, 117
LCD	Liquid Crystal Display	16
LDI	Layered Depth Image	46, 47
LDV	Layered Depth Video	1, 47
LFN	Light Field Node	104–107, 111, 118, 121, 130, 183, 185, 186
LLF	Layered Light Field	46
MSE	Mean Squared Error	36, 37
NCP	Near Clipping Plane	28, 29, 40, 148–150, 164, 165
NDC	Normalised Device Coordinates	24, 25, 29, 30, 32, 33, 41

OpenGL	Open Graphics Library	vii, ix, 24, 26, 27, 30, 33, 38–40, 42, 46, 54, 148, 150, 160, 161, 164, 167–171, 185
OpenMP	Open Multi-Processing	111, 183–185
PMD	Photonic Mixer Device	46
PSNR	Peak Signal to Noise Ratio	16, 36, 37, 68, 76–79, 81–83, 90–97, 99, 128, 137–141, 151–158
RAM	Random-Access Memory	165
SIMD	Single Instruction Multiple Data	26, 39, 50
SLERP	Spherical Linear Interpolation	70

List of Figures

List of Tables

Chapter 1

Introduction

In the last few years, a couple of novel displays emerged, allowing for a glass-free 3D viewing experience of both video and motionless poster content. In contrast to stereoscopic two view displays, which become increasingly popular in the home consumer market, several autostereoscopic display types are in development, ranging from an early to a late development stage. Advancements in the display technology allowed for a rapid grow in the number of views of those autostereoscopic displays, which comes along with a vastly increasing data rate that has to be processed and stored.

Hence, concepts developed for stereoscopic two view displays, whose content is typically provided in form of a two view video stream, or as video plus depth, can often not be applied to autostereoscopic displays. Although, for the first commercially available autostereoscopic multi-view displays the Layered Depth Video (LDV) format is used (see Müller et al. [MSD+08]), emerging display technologies demand for rendering concepts and input formats tailored to the special requirements of these displays.

The application of autostereoscopic multi-view displays is mainly in advertisement, where the observer's position can not be controlled, and the application of special headgear is not feasible. As typical in advertising, the scenes shall meet highest quality standards and are normally rendered via ray tracing. The creation of artificial content can be a complex work flow, composed of the creation of the 3D models, arranging a scene, adding lighting, effects, and animation to the scene, rendering of the images, and post processing of the images.

For each media there are certain aspects to be taken into account for content creation, e.g. creation of images for poster advertisement, television, and printed media. The same holds for content creation for stereoscopic displays, which adds another dimension to the conventional 2D content generation process. For example, the scene has to be planed more carefully with regards to the observer's position, as objects that are placed in front of

the display will destroy the stereoscopic perception of the scene when they collide with the border of the display. The production of 3D content is already a costly and time consuming process, hence, another requirement is that the work flow of the content creation for autostereoscopic multi-view displays should be similar to that of traditional 3D models used in advertisement. One example is the content of an autostereoscopic poster display whose content is rendered with over 200,000 views per pixel. Rendering of the full data set of such a display takes several month per square meter on a single workstation. This massive amount of data imposes a challenge to the rendering process, the data transfer, and the data storage.

Other challenges are related to autostereoscopic multi-view video displays. Although, the number of views are relatively low, when compared to the example of the poster display, a smooth playback of the video content requires at least 20 Frames Per Second (FPS). Such data rates exceed by far the specification of today's consumer hardware and require a large compression factor, to be economically feasible transferred.

The introduction of compression requires to decompress the data on the device. Due to the vast amount of data, the challenge is to find a decompression algorithm that can be parallelized and is computationally inexpensive, in order to achieve real-time for smooth playback.

Summarising, the used interpolation algorithm should effect the content creation as little as possible and should integrate well with the available modelling tools on the market. At the same time, the rendering costs for ray tracing of the content should be reduced. For video displays, a compression has to be implemented to reduce the data rate to a manageable amount, as well as a decompression algorithm that can be implemented on the device and achieve real-time.

This work evaluates the hypothesis that Depth Image-Based Rendering (DIBR) can meet these challenges. The input data of DIBR algorithms are colour and depth images, which are both readily available by virtually every modelling tool, and therefore, do not interfere with the content creation process. By ray tracing only a small amount of the colour and depth images, DIBR is used to exploit the correlation between the images and interpolate the full data set. This reduces the amount of data in the content creation step, avoiding rendering, transfer, and storage of the full data set. Finally, a computationally inexpensive DIBR algorithm is to be found that allows for real-time decompression on the device, without introducing prominent artefacts.

Outline of the Thesis

The main contribution of this work is the development of different DIBR algorithms for autostereoscopic multi-view poster and video displays. Starting with an overview of 3D display technologies in Chapter 2, the focus is then shifted to integral imaging and the representation of light fields in the context of standard camera models, as well as rendering techniques for autostereoscopic full parallax multi-view displays. Afterwards, an overview of the related literature is given and the basics for image interpolation is introduced, along with the used error metrics. In order to get an estimate and set a goal for data processing, the data rate for autostereoscopic multi-view video displays is derived from the literature. Special care is taken to address the relation between the technical specification of a multi-view display and the derived requirements for content creation and view interpolation.

The data sets are introduced in Chapter 3 along with a ray based simulator for multi view displays. The simulator is utilised to render virtual views of a simulated display, allowing for an evaluation of the interpolated images that is more relevant to a potential observer.

The subsequent Chapters 4 and 5 introduce and evaluate basic interpolation algorithms for perspective and orthographic images.

A novel DIBR algorithm will be designed in Chapter 6, built on a point based scene representation, to interpolate the complete data set. For 3D posters, the interpolation does not have to be in real-time but should offer a computational advantage, compared to the rendering of all images. In order to hold a maximum amount of input data in memory, a lossy compression scheme will be utilised, allowing to omit redundant data of Lambertian surfaces.

The scene representation is also exploited for a scene analysis, in order to compute an optimal set of input images for the interpolation. This challenge will be met with a Best-Next-View (BNV) selection algorithm (Chapter 7), tailored to the special requirements of a multi-view display, that follows the generate-and-test paradigm and delivers a ranking of the potential viewpoints, ordered by their importance regarding selected properties. This ranking is used to select an optimised set of input images, before the costly rendering of the colour images with the ray tracer is initiated.

Finally, a novel interpolation scheme is presented in Chapter 8, which combines perspective and orthographic input data in order to improve the

quality of DIBR results for autostereoscopic full parallax video displays. The different properties of the two image representations will be exploited to greatly reduce the amount of input images, without introducing prominent artefacts that disturb the perception of the scene. The goal is to reduce the data rate by an amount that allows transfer to the device with off-the-shelf hardware. A special requirement is that the interpolation algorithm can not resort to a scene analysis, in order to achieve real-time for playback of the content.

Contribution

The main contributions of this thesis are different DIBR algorithms, developed for autostereoscopic poster and video displays. Some parts of this thesis have been previously published as peer reviewed articles.

- Daniel Jung and Reinhard Koch. Efficient Depth-Compensated Interpolation for Full Parallax Displays. In 5th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), 2010. [JK10]

- Daniel Jung and Reinhard Koch. Efficient Rendering of Light Field Images. In Daniel Creamers, Marcus Magnor, Martin R. Oswald, and Lihi Zelnik-Manor, editors, Video Processing and Computational Video, volume 7082 of Lecture Notes in Computer Science, pages 184–211. Springer Berlin Heidelberg, 2011. [JK11b]

For autostereoscopic multi-view poster displays, a DIBR algorithm is introduced that incorporates a point based scene representation. In order to fit into memory, the view-dependent colour information of the surface is compressed by a lossy encoding, which exploits the similarity of Lambertian surfaces. The algorithm is efficiently accelerated by utilising multi-core Central Processing Units (CPUs) and the Graphics Processing Unit (GPU). This work is introduced in Chapter 6, and some parts in more detail in the Appendix, which has previously been published in [JK10] and [JK11b].

- Daniel Jung and Reinhard Koch. A Best-Next-View-Selection Algorithm for Multi-View Rendering. In 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011. [JK11a]

- Daniel Jung and Reinhard Koch. Efficient Rendering of Light Field Images. In Daniel Creamers, Marcus Magnor, Martin R. Oswald, and Lihi Zelnik-Manor, editors, Video Processing and Computational Video, volume 7082 of Lecture Notes in Computer Science, pages 184–211. Springer Berlin Heidelberg, 2011. [JK11b]

A key factor for the fidelity of the image interpolation is the selection of the input images. A BNV selection algorithm is introduced in Chapter 7, which computes a ranking for the set of input images, based on an analysis of the geometry of the scene. This allows to select the input images for DIBR in dependence of the scene, and helps to avoid artefacts that disturb the perception of the scene. The work has been previously published in [JK11a] and [JK11b].

- Daniel Jung and Reinhard Koch. Image Based Rendering from Perspective and Orthographic Images for Autostereoscopic Multi-View Displays. In International Workshop on Vision, Modeling, and Visualization, VMV, pages 187–194, 2013. [JK13]

Chapter 8 introduces a DIBR algorithm that is targeted to be implemented on a Field-Programmable Gate Array (FPGA), located at the video device. Hence, the data rate is drastically reduced, because only the input images of the DIBR have to be transferred to the device, allowing to operate an autostereoscopic multi-view video display with off-the-shelf hardware. The foundation for the algorithm is laid in the Chapters 4 and 5, and has been previously published in [JK13]. Also, some minor parts of Chapter 2 have been previously published in [JK10], [JK11a], and [JK11b].

Related Peer Reviewed Publications

The interest in GPU accelerated real-time applications lead to the following contributions that show the potential of General-Purpose computing on Graphics Processing Unit (GPGPU) in applications other than rendering but are not discussed explicit in this thesis.

- Bogumil Bartczak, Daniel Jung, and Reinhard Koch. Real-Time Neighborhood Based Disparity Estimation Incorporating Temporal Evidence. In Proceedings of the Deutsche Arbeitsgemeinschaft für Mustererkennung, DAGM-Symposium, pages 153–162, 2008. [BJK08]

Per frame based real-time depth estimation for stereo video sequences ignores an important resource, the temporal coherence between adjacent video frames. Under the assumption of small changes between adjacent frames, caused by camera and object movement, the idea pursued in [BJK08] is to reuse the dense disparity estimation of the current set of frames in the subsequent set of input images, in order to guide the depth estimation and achieve a temporal smoothing, guided by strong evidence, which is built and distributed along the timeline.

The coherence between temporal adjacent frames are estimated by a dense optical flow estimation, in order to compensate camera motion and object movement between the frames.

▶ Anne Jordt-Sedlazeck, Daniel Jung, and Reinhard Koch. Refractive Plane Sweep for Underwater Images. In German Conference on Pattern Recognition, GCPR, pages 333–342, 2013. [JSJK13]

The estimation of dense disparity using the plane sweep algorithm is known to be an excellent application for the implementation on the GPU, as it maps well to the hardware and can be implemented efficiently. In [JSJK13] the plain sweep algorithm is adapted to handle arbitrary ray-based camera models for the application of underwater depth estimation, which accounts for refraction at the air, glass, and water interface.

Chapter 2

Basic Principles

The transition from conventional single view displays over stereoscopic two view displays to autostereoscopic multi-view displays is accompanied by an enormous increase of the data rate, depending of the number of supported views. In the later, the different display technologies will be introduced, in order to distinguish the properties of autostereoscopic full parallax displays, and to motivate the benefits, these displays have in some applications. But foremost, the basis of mono- and binocular human depth perception is introduced, which is the foundation for stereoscopic display devices.

2.1 Human Depth Perception

Humans derive a visual three dimensional impression of a scene, based on the interpretation of images retrieved from monocular and binocular vision. The effect that the perceived size of an object is constant when the distance changes is referred to as *size constancy*. Holway and Boring [HB41] did experiments in order to explain size constancy by reducing the observer's perception gradually to the retinal image, where the perception of size solely depend on the *law of the visual angle*. Their results show that size constancy can also be achieved with monocular vision. It was only after removing accommodation and the visual frame that the perception of size agreed with the law of the visual angle.

Wallach and Zuckerman [WZ63] demonstrated that under certain conditions monocular depth cues prevail over depth perceived by binocular vision. They used anaglyphs and a pseudoscope to create a contradiction between binocular depth and perspective depth cues. In the presence of patterns that created strong perspective distance cues, the perspective depth cues prevailed over the binocular ones, whereas in absence of perspective cues the depth perception was inverted.

In the literature, different opinions about the number of properties in the interaction of the human eye with the world, which deliver depth clues, are found, e.g. Okoshi [Oko76] and Cruz-Neira et al. [CNSD93]. In the following, eight different depth cues are introduced that are commonly regarded as most significant. Several depth clues can be derived from monocular vision and have been used for a long time in paintings to create believable images.

Occlusion or interposition is based on the assumption that light travels in a straight line and objects in the scene does not let light pass through. In addition, an a priori knowledge about the objects is required to distinguish an occluded object from its occluder. When those conditions are fulfilled the interpretation is that the occluder is nearer to the observer than the occluded object.

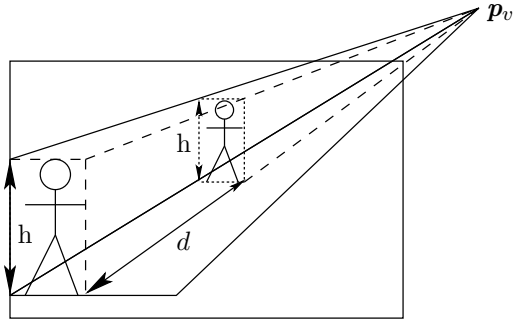

Figure 2.1. Relative distance of two identical objects under a perspective projection.

Linear perspective is based on the observation that under perspective projection all parallel lines that are not perpendicular to the image plane converge to one finite vanishing point $\boldsymbol{p}_v$ in the image plane (Hartley and Zisserman [HZ00]). Consequently, any object that can be enclosed by parallel lines converges to the vanishing point of these parallel lines, when moved along the direction of the enclosing parallel lines. Hence, an observer with knowledge about the relative size of the depicted objects h can infer the relative distance d of the objects by its size, and parallel lines that are present in the scene (cf. Figure 2.1).

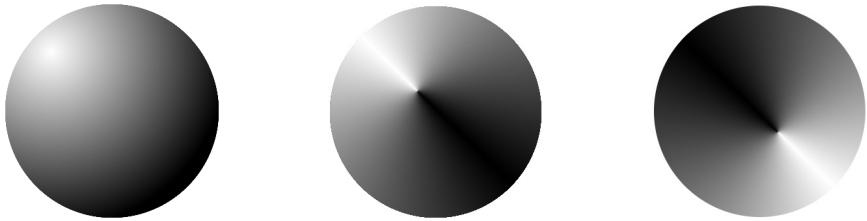

Figure 2.2. Under the assumption of a directional light source in the upper left the shaded circles can be interpreted as a ball (left), a conus (centre), and a funnel (right).

Lighting and shadows provide important clues about the 3D shape of an object. Figure 2.2 shows three projections of identical shape, distinguishable only by different shade. Diffuse shading depends solely on the direction of the light source and the surface normal. Hence, with the assumption of a light direction and knowledge about the principles of diffuse shading the surface normals can be derived. The right and the centred objects in Figure 2.2 are identical, except that the right image is rotated by 180 degree. Assuming the light source is in the upper left the interpretation of the centred object is a conus and that of the right object is a funnel. If the assumption of the direction of the light source changes to the lower right, the derived normals are inverted, leading to a reinterpretation of the shading and a different 3D shape.

Atmospheric effects summarise all volumetric effects where the colour of a light ray is modified in dependence of the distance travelled through a medium, converging to a basic colour for an infinity distance. This volumetric effect applies for example under foggy atmospheric conditions or underwater.

Motion parallax manifests as the disparity between two consecutive images of a sequence. For a perspective camera, solely translated by a fixed amount, the amount an object is translated in the image, the disparity δ, depends on the distance of the object to the camera. The closer an object is to an observer the larger is the disparity induced by the viewpoint motion, converging to zero for an object at infinity when no rotation is

present. *Motion pictures* therefore add another potential depth clue that is not available in still images. In addition motion pictures allow to expose occluded regions, allowing for a better perception of the scene and of 3D objects.

Convergence describes the process of directing the fovea centralis of both eyes to a point of interest in space. Around that area of interest, the disparity between the images of both eyes is zero, effectively allowing for a consistent perception.

Binocular disparity originates from the parallax between the eyes of an observer. Based on an analysis of the differences in the images presented to each eye convergence can be achieved, and the visual perception allows for an illusion of depth. *Stereoscopic two view displays* generate this illusion by offering a second image from a different viewpoint, a separate image for each eye. Displays without tracking typically have a fixed baseline and assume a fixed viewpoint, therefore the motion of an observer or the distance to the display is not compensated. This limitation of two fixed viewpoints is the source of a couple of contradictory depth cues the observer may perceive, leading to discomfort and an inconsistent depth illusion, especially noticeable when the observer starts to move in front of the display. In contrast, the images offered by tracked stereoscopic two view displays or autostereoscopic displays depend on the observer's position.

Accommodation is the ability to focus the eye on objects in varying distances. In natural viewing the eyes are generally focused on the point of convergence but graphics systems usually ignore accommodation clues, rendering everything in focus [CNSD93]. When viewing a 3D scene on a stereoscopic display, the eyes focus on the display device, which adds to the potential irritating all in focus rendering.

2.2 Classification of Display Technologies for 3D Content

When an observer looks at a window both can be seen, objects that are in front of the window and objects that lie behind the window. Given the

viewing rays were fixed at the surface of the window and at the observer's position, the observer can sample the world in front and behind the window by a change of the position. Now consider a single light ray, then the angle of that ray to its fixed surface point of the window will change according to the observer's position and the colour of that light ray will change according to the surface position that is sampled, and the angle of incidence. If all light rays that pass through that window could be recorded and properly displayed by a display, this would be an ideal autostereoscopic full parallax display. An observer of the display would see no difference between the glass window and the display and each eye of the observer would see a different image of the scene in a most natural way.

Technical realisations of autostereoscopic displays differ from the ideal display in several aspects. The resolution and the number of supported views are both limited and the viewing angle is usually less than 180 degree, depending on the display technology. This adds up with limitations of the colour space and limitations of the luminance of the display technology.

2.2.1 Nomenclature

Conventional displays that are designed for 2D content are often described by their resolution and size. This classification alone is not sufficient in the context of stereoscopic displays because depending on the display technology, e.g. the resolution might or might not be divided by the number of views, leading to a lower resolution of a single view. Therefore it is not possible to derive meaningful technical specifications, like Dots Per Inch (DPI), from the resolution and display size alone, without detailed knowledge of the display technology. In order to compare different stereoscopic display technologies the following descriptors are derived.

Spatial Resolution of a Display

The *spatial resolution* $R_{S,R(v)}$ is defined by dividing the metric size of one view S_v by the resolution in pixels of one view R_v

$$R_{S,R(v)} := \frac{S_v}{R_v}, \tag{2.2.1}$$

given in width and height. The resolution of one view is identical with the well known resolution of a single view display with respect to a single

eye. This allows for a meaningful derivation of technical specifications like DPI in the context of 3D content. Another benefit is that the spatial resolution can be directly compared with the resolution of a single view display, which might help to get a meaningful idea of the capabilities of the display. The spatial resolution is not necessarily a fixed value. Some displays have different spatial resolutions for the display of 2D and 3D content.

Angular Resolution of Autostereoscopic Displays

Besides the spatial resolution, autostereoscopic displays have other technical specifications of interest, the number of views and the viewing angle they are distributed in. The *angular resolution* R_a of a display is defined as the angle between two neighbouring views. For autostereoscopic displays that distribute the views equiangular in the viewing angle, the angular resolution is given by

$$R_a := \frac{\Psi}{\nu}, \tag{2.2.2}$$

dividing the maximum horizontal viewing angle Ψ, in which the views are distributed in, by the number of horizontal views ν. A high angular resolution will allow for a smooth view transition, when the observer enters another view by movement, and will allow an observer to move far away from the display without loosing the 3D impression. A single view is defined as one viewing direction, hence all parallel rays of the same direction are part of one view. The set of all views T are given by the number of horizontal views ν and the number of vertical views ξ by

$$T := \left\{(a,b) \mid a_{\in}\mathbb{N}_{<\nu}, b_{\in}\mathbb{N}_{<\xi}\right\}. \tag{2.2.3}$$

For stereoscopic two-view displays, this technical specification will have no meaning, because the shown views are from a fixed viewpoint, regardless of the observer's position.

2.2.2 Stereoscopic Two View Displays

According to Okoshi [Oko76], and Johnson and Jacobsen [JJ05], the first attempt to draw stereoscopic images had been undertaken by Giovanni Battista della Porta and dates back to the early 17^{th} century. Later, Wheatstone [Whe38] laid the foundations of stereoscopic imaging. He discovered

that two images mimicking the different viewpoints of an observer's eyes could be used to gain a three dimensional impression of the scene, when the images were exclusively displayed to the corresponding eye. Wheatstone constructed the first stereoscope, which directed two on canvas painted views separately to the observer's eyes, using mirrors. This basic principle is used in all stereoscopic displays but the view separation is achieved by different methods.

Stereoscopic two view displays have been available for a long time. Today, displays that are available for the consumer market are mostly based on conventional display devices, combined with a filter for view separation. The filtering is either applied in the colour domain, in the temporal domain or by the orientation of the electric field vector, in case of polarised light.

The application of colour filters to achieve a stereoscopic view dates back to Rollmann [Rol53]. Rollmann used the colours blue and yellow to draw stereoscopic views, and red and blue filter glasses to separate the views. According to Abramson [Abr03] the first working colour and stereo television system was reported to be developed by J. L. Baird in 1941. However, for the consumer market it took more than a decade until coloured television systems became available (Butler [But06]), allowing for the broadcast of anaglyph 3D content, viewed through anaglyph filter glasses.

Stereoscopic shutter systems show alternating images targeted at the left and the right eye. In combination with shutter glasses, synchronised with the display device, the eye of the observer, for which the image is currently not displayed, is occluded. Due to the filtering in the temporal domain, stereoscopic shutter systems have to operate at least with twice the frame rate of the content that is displayed, but often operate with the fourfold frame rate to eliminate the perception of flickering.

Stereoscopic displays based on linear polarised light emit both views with a different orientation of the electric field vector. The views are separated by filter glasses that match the different orientations, allowing the matching orientation to pass through and filtering the orthogonal orientation. Systems based on circular polarised light add the benefit that the observer does not need to keep the glasses aligned with the display, allowing for an unrestricted head movement with constant view separation.

Untracked stereoscopic two view displays have two major limitations. First, the observer needs to wear headgear for view separation that is tuned to the display device. The other limitation is that only two views from a fixed viewpoint are displayed. If the observer moves, the displayed stereoscopic

images does not change accordingly, which causes conflicting depth cues. Hence, the observer's movement has to be restricted to allow for a believable 3D perception. This limits the application of stereoscopic two view displays to theatres or home application. The display of stereoscopic content in public places, e.g. for advertisement, demands for another technology that does not require a special preparation of the audience.

2.2.3 Autostereoscopic Displays

Autostereoscopic displays allow to display stereoscopic images without the need for headgear. The observed images depend on the position of the observer, therefore an observer's movement will not result in conflicting depth cues as long as the observer stays inside the viewing angle of the display.

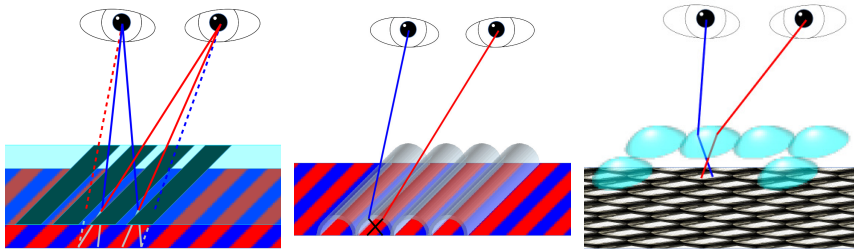

Figure 2.3. Sketch of a parallax barrier display (left), a lenticular lens based display (centre), and a display based on a lens array (right). The views for the left eye (blue) and the view for the right eye (red) are aligned as stripes behind the lenticular lens and the parallax barrier, respectively. Behind the lens array, a full 2D image allows for a full parallax inside the viewing angle of the display.

Ives [Ive03] invented a way to capture and display stereoscopic images based on *parallax barriers*, allowing for two views. His invention allowed for an autostereoscopic view with horizontal parallax but restricted the viewer to a small viewing area. His idea was to place a layer with alternating transparent and non-transparent stripes in a given distance to a screen, see Figure 2.3 (left). The transparent stripes would be small in relation to the non-transparent stripes, dividing the image information up into information for the left and right eye by occlusion (stippled and continuous lines in

Figure 2.3, left). Using holes instead of transparent stripes, this principle allows to build a relatively simple autostereoscopic full parallax display, utilising a high resolution screen. The non-transparent layer with the small transparent holes are mounted with a given distance to the screen. With a small size of the transparent holes, each would act as a small pinhole camera, projecting a full 2D image. The drawbacks of such an autostereoscopic device is that the resolution of the display is divided by the number of offered views and the small size of the pixels in relation to the non-transparent part of the occluding layer, the aperture mask.

In 1908 Lippmann [Lip08] invented *integral imaging*. He had the idea to enhance photography by offering a free viewpoint image, allowing to explore occlusions by viewpoint shift and full parallax autostereoscopic images, analogue to the view out of a window. His idea was to place an array of lenses in front of a photo film and record the inclining light through the lenses onto the film (Figure 2.3, right). After processing and fixing the film, he proposed to backlit the exposed film by a diffuse light source, projecting the captured scene back into space. Autostereoscopic full parallax devices based on lens arrays have similar properties compared to parallax barrier displays but allow for larger apertures and therefore larger pixel sizes within the aperture mask.

Typical *lenticular lens* based autostereoscopic systems offer only one parallax and can be thought of as a simplified lens array, reducing the parallax to one dimension. In 1922 Curwen [Cur23] described a device based on lenticular lenses that could display different images, based on the viewpoint or by moving parts of the device. Figure 2.3 (centre) shows a sketch of a lenticular based autostereoscopic device. The lenses have the form of a one layer stack of half-cylinders. Parallel to the cylinder's axis the displayed content does not depend on the viewpoint, therefore the image content behind the lens system belongs to one view only. Orthogonal to the cylinder's axes, the displayed image content depends on the viewpoint, hence the different views that are displayed are encoded as aligned stripes, parallel to the cylinder's axes. Lenticular based systems reduce the horizontal resolution by the number of available views but contrary to lens array based displays, it is constructed without an aperture mask, allowing for full sized pixels.

The autostereoscopic displays from above all use a 2D screen for image formation in combination with lenses or an aperture mask for view separation. *Volumetric displays* extend the image formation by one dimension, allowing

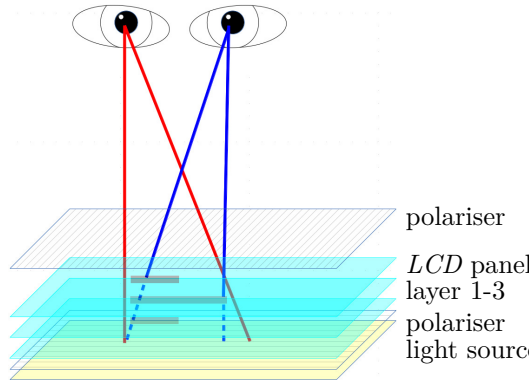

Figure 2.4. Sketch of a volumetric multi-layer *LCD*. At each layer a 2D image is displayed by the *LCD* panel matrix. The superimposed images within the enclosed volume form a full parallax autostereoscopic display.

the modification of light within a physical volume. Leung et al. [LIE98] invented autostereoscopic displays based on a multi-layer volumetric Liquid Crystal Display (LCD). At each panel layer of the display, the orientation of the polarized back light can be changed. Hence, the resulting intensity of each light ray depends on the intensity modulation at each panel layer along the line of sight through the stack of panel layers of the display. This allows for an autostereoscopic multi-view display with a large viewing angle (see Figure 2.4). Putilin et al. [PLK01] used a multi-layer LCD with a neuronal network for image processing. According to Wetzstein [Wet11], the benefits of polarisation fields improve the spatial resolution, brightness, and depth of field, compared to parallax barriers and integral imaging, by maintaining thin form factors. The limitations of multilayer polarisation fields, as described by Wetzstein, are the requirement of correlated views and the convergence to a moderate Peak Signal to Noise Ratio (PSNR), from which Wetzstein concluded that polarisation fields may have a limited degree of freedom. For a more comprehensive overview of volumetric displays, and the history of volumetric devices, refer to Favalora [Fav05].

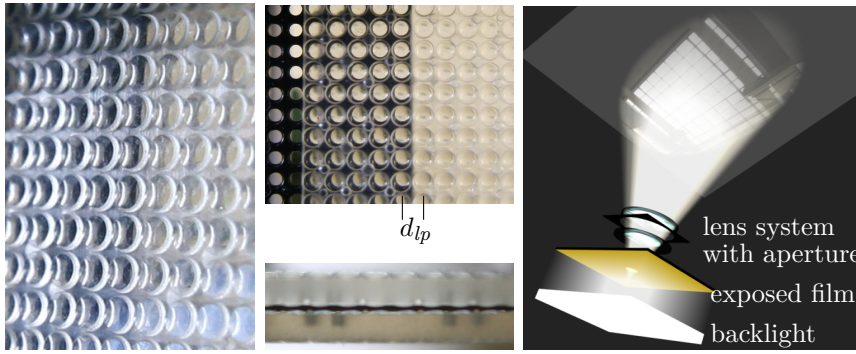

Figure 2.5. Photo of the lens system composed of two lens arrays and an aperture in between (left and centre, $d_{lp} = 2$ mm). The right hand side shows a schematic of one pixel of the display.

2.3 Technical Realisation of Lens Array Based Displays

Full parallax lens array based autostereoscopic displays are based on the work and insights of Lippmann [Lip08].

One technical realisation of the principle is depicted on the right hand side of Figure 2.5. It shows one pixel of the display that works similar to a film projector. The view dependent colour information is stored on an exposed film that is lit from behind. In front of the film is a lens system that allows for a separation of the view dependent colour information. The lens system is modelled by a spherical camera, which allows for an equiangular mapping of the viewing rays to a 2D image. The image of a lens system is called an *elemental image*. A paper is added to the scene to show the projected image. An observer of the pixel will only see a part of the projection, depending on its position. The technical realisation shown in Figure 2.5 allows for a full parallax, i.e. the stereoscopic effect experienced by an observer in front of the display does not depend on the horizontal alignment of the observer's eyes with the display.

The left image of Figure 2.5 shows a photo of the lens system consisting of two lens arrays and an aperture in between. The distance between two

neighbouring lens systems, the lens pitch d_{lp}, is drawn in the top image of the centre of Figure 2.5, which is a top view of the lens system. For displays based on lens arrays, the lens pitch equals the spatial resolution ($R_{S,R(v)}$) derived in Equation 2.2.1. The bottom image shows a lateral cut of the lens system with the black aperture in between the lens arrays. In the following, autostereoscopic displays will be assumed with a full parallax, based on the design of Figure 2.5 with a lens array in front of an integral image.

2.3.1 Representation of Light Fields

Early work of Weber [Web85] (1885) focused on the intensity measuring of diffuse daylight using translucent glass. Weber was already aware, that the illumination distribution for a single point has to be described by a brightness value for every direction in space. Moon and Timoshenko translated Gershuns [Ger36] work about the introduction of light fields from a geometrical point of view. In Gershuns work about the fundamental theory of light fields, he described Webers representation of the brightness-distribution solid as the "most complete description of the light field", which essentially associates a brightness value to every direction in space for a given point. This representation already allowed for a complete description of a light field of static scenes.

Adelson and Bergen [AB91] introduced the plenoptic function. The plenoptic function can be used to describe all light that exists within a scene. The idea is to place a sphere at every possible position ($\mathbf{V} = [\,\boldsymbol{V}_X,\, \boldsymbol{V}_Y,\, \boldsymbol{V}_Z]^T$) and to record all light rays passing through the spheres centre at the angles (φ and θ). In addition, the function parametrises the wavelength λ and the time t. The plenoptic function has therefore the form

$$P = P(\varphi, \theta, \lambda, t,\, \boldsymbol{V}_X,\, \boldsymbol{V}_Y,\, \boldsymbol{V}_Z) = P(\varphi, \theta, \lambda, t,\, \boldsymbol{V}). \tag{2.3.1}$$

Under the assumption that the scene is static and constantly illuminated, the plenoptic function is reduced to the 5D function

$$P_{5D} = P_{5D}(\varphi, \theta,\, \boldsymbol{V}_X,\, \boldsymbol{V}_Y,\, \boldsymbol{V}_Z) = P_{5D}(\varphi, \theta,\, \boldsymbol{V}). \tag{2.3.2}$$

This reduced form can be considered as a full panoramic image taken from position $\mathbf{V}$. McMillan and Bishop [MB95] discussed the representation of a plenoptic sample and came to the conclusion that a unit sphere centred

at **V** would be the most natural representation, but they chose a cylindrical projection because of the simplified correspondence search and the possibility to easily unroll it onto a planar map.

Figure 2.6 shows the viewing frustum of a single lens system of a full parallax display, based on lens arrays, with the display plane represented by a quad in its centre. Relating the viewing data of a lens array based

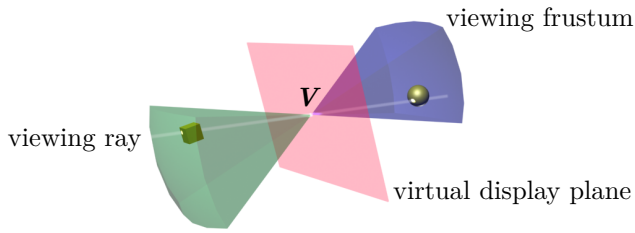

Figure 2.6. The viewing frustum of one lens system of a lens array based full parallax display.

display to the plenoptic function (see Eq. 2.3.2) the position of the projection centre of one lens system is encoded as position **V** on the display plane (see Fig. 2.6), the polar angle φ is restricted to the maximum viewing angle $\mathbf{\Psi}$ of the display with the zenith perpendicular to the display plane, and the azimuthal angle θ is restricted to the interval $[0, 2\pi[$. For digital image processing, the resulting intensity is encoded according to the RGB colour model or in greyscale. For time variant autostereoscopic multi-view video displays the time t is parametrised according to Eq. 2.3.1. The viewing frustum is sampled with the angular resolution of each lens system, therefore the sampling of the scene is related to the number of views T of the display (see Eq. 2.2.3). The set of viewing directions T is mapped to the incidence angle in spherical coordinates Υ by

$$\Upsilon := \{(a, b)_{\in}\mathbb{R} \times \mathbb{R} \mid 0 \leqslant a \leqslant \pi, 0 \leqslant b < 2\pi\} \tag{2.3.3}$$

$$A : T \mapsto \Upsilon. \tag{2.3.4}$$

Further,

$$B : \mathbb{N}^2 \mapsto W. \tag{2.3.5}$$

maps each lens system of the display $(\cdot, \cdot)_{\in} \mathbb{N}_{>0,\leqslant R_v} \times \mathbb{N}_{>0,\leqslant R_v}$ to its 3D position in the world coordinate system $W_{\in}\mathbb{R}^3$.

The viewing frustum of the multi-view display is then defined as the union of the viewing frusta of all lens systems of the display. Hence, the time variant light field LF of the display is the union of the view dependent colour information of all lens systems of the display and is described by

$$LF = \bigcup_{\boldsymbol{V} \in B(R_v)} P_p\left(\varphi, \theta, \lambda, t, \boldsymbol{V}\right). \tag{2.3.6}$$

For displays based on lens arrays, the data is stored as 2D image data for each lens system of the display. In order to display arbitrary computer generated content, in the following the process of image formation in computer graphics is described and linked to the light field of autostereoscopic displays.

2.4 Image Formation in Computer Graphics

The light fields required by autostereoscopic displays are stored as 2D image data, encoding the light rays from a 3D scene, passing through the virtual image plane of the display. In computer graphics, 2D images are usually generated from a 3D model. A scene is modelled by composing predefined objects. Objects are defined pointwise by vertices, a group of vertices form a polygonal surface, and a group of surfaces form the surface of an object.

Besides the geometry of the scene, lighting and properties of the surfaces of objects can be altered, e.g. the colour, the light interaction properties, texture, and transparency. In general, the images are afterwards computed by adding a camera to the scene and by processing the 3D model. Before the coordinate systems and camera models are introduced, the representation of 2D images are formalised.

2.4.1 Image Representation

Images processed by a computer are usually stored as two dimensional data arrays. The discrete picture elements (pixels) of Charge-Coupled

Device (CCD)-sensors in electronic cameras deliver spatial discrete intensity readouts on the input and display devices allow to control the discrete pixels of the output screen. An image is therefore defined as a 2D matrix of pixels, with a set of columns

$$U = \left\{ u : u \in \mathbb{N}_{[1,U']} \right\} \; ; \; U' = \text{number of horizontal image pixels} \tag{2.4.1}$$

and a set of rows

$$V = \left\{ v : v \in \mathbb{N}_{[1,V']} \right\} \; ; \; V' = \text{number of vertical image pixels} \tag{2.4.2}$$

resulting in a $|U| \times |V|$ 2D position matrix. For some applications, more than one date has to be stored per pixel position, therefore the 2D position matrix is expanded by

$$c = \left\{ c : c \in \mathbb{N}_{[1,c']} \right\} \; ; \; c' = \text{number of image channels,} \tag{2.4.3}$$

allowing to store additional dates for each 2D pixel position within the image. For colour images the commonly used 24-Bit RGB-colour-model is assumed, encoding the light intensities for the colours red, green, and blue for each pixel in a separate image channel with 8-Bit. Therefore, a colour image can be described as a function

$$I : U \times V \times c \rightarrow \mathbb{N}_{\leqslant 255} \tag{2.4.4}$$

assigning each 2D pixel position and channel in the image an 8-Bit integer value. Disparity images encode distances instead of colour data for each pixel of an image, to a given reference image. Assuming that two channels are used for the horizontal and vertical disparity, a disparity image is described as a mapping

$$D : U \times V \times \{0, 1\} \rightarrow \mathbb{R}, \tag{2.4.5}$$

that assigns a disparity value to each 2D pixel position in the image. Similar, depth images are defined as a mapping

$$D : U \times V \rightarrow \mathbb{R}, \tag{2.4.6}$$

assigning a single depth value to every 2D pixel position. Disparity and depth values are both encoded as 16- or 32-Bit floating point numbers.

2.4.2 Homogeneous Coordinates

In Computer Graphics, homogeneous coordinates are used because they have properties that reduce the complexity to describe points at infinity, projections, and concatenate affine transformations. Given a point $\boldsymbol{p} = [x, y, z]^T_{\in}\mathbb{R}^3$ in Euclidean 3-space its homogeneous coordinates are defined as $[x \cdot w, y \cdot w, z \cdot w, w]^T$ with $w_{\in}\mathbb{R}_{\neq 0}$. Canonical, Euclidean points are transferred into homogeneous coordinates with $w = 1$, resulting in $[x, y, z, 1]^T$ for $\boldsymbol{p}$.

Homogeneous coordinates are transformed into Euclidean coordinates by scalar division of the vector by its 4^{th} component, resulting in the canonical form, and removing the last coordinate.

Points at infinity, which can not be numerically computed in Euclidean 3-space, are described in homogeneous coordinates by setting the 4^{th} component to zero. Another benefit of homogeneous coordinates is the elegant description of the non-linear perspective division of the perspective projection.

In the next section a world coordinate system is established and the property of homogeneous coordinates to concatenate affine transformations is utilised to relate different coordinate systems.

2.4.3 Coordinate Systems

In order to arrange a scene and render images, different tasks have to be fulfilled. Objects have to be created and arranged to form a scene, a camera has to be placed, and the output image has to be computed. Before the rendering process is described in detail, the different coordinate systems that are involved are introduced, and the relationship between them.

World coordinates define a world coordinate system W against that all other coordinate systems are registered (see Figure 2.7).

Camera coordinates (also called eye coordinates) refer to the camera centre as origin. The orientation of the camera coordinates C often align an axis of the coordinate system with the optical axis of the camera. The camera coordinate frame is transformed by the *viewing transformation* $\mathcal{V}^{-1}$

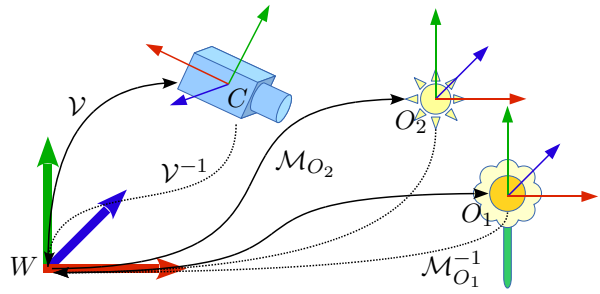

Figure 2.7. Camera C and object coordinates O in relation to the world coordinate system W.

into world coordinates, according to

$$\mathcal{V} = \begin{pmatrix} \mathcal{R}_C & \boldsymbol{t}_C \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}; \mathcal{V}^{-1} = \begin{pmatrix} \mathcal{R}_C^{\mathrm{T}} & -\mathcal{R}_C^{\mathrm{T}}\boldsymbol{t}_C \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}. \tag{2.4.7}$$

In relation to the world coordinate system, the camera is positioned by a 3D translation vector $\boldsymbol{t}_C$ and aimed by the 3D rotation matrix $\mathcal{R}_C$.

Object coordinates are helpful when an object is modelled. Vertices are positioned relative to the local object coordinates O, detached from later application. The object coordinate frame is transformed by the *modelling transformation* $\mathcal{M}^{-1}$ into the world coordinate system by

$$\mathcal{M} = \begin{pmatrix} \mathcal{R}_O & \boldsymbol{t}_O \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}; \mathcal{M}^{-1} = \begin{pmatrix} \mathcal{R}_O^{\mathrm{T}} & -\mathcal{R}_O^{\mathrm{T}}\boldsymbol{t}_O \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}. \tag{2.4.8}$$

For rendering, the object coordinates are transformed into camera coordinates by first applying the inverse modelling transformation, followed by the viewing transformation

$$\mathcal{Q} = \mathcal{V}^{-1}\mathcal{M}. \tag{2.4.9}$$

Normalised Device Coordinates (NDC) are used between the projection and the viewport transformation. The different cameras introduced in the later will transform a part of the scene into the cubic interval of

$[-1, 1]$, which defines the part of the scene that is used for image formation. Hence, regardless of the different projections, the vertices can be uniformly processed after transformed into Normalised Device Coordinatess (NDCs).

Image coordinates (or window coordinates) are obtained by transforming the first two components of the vertices from NDC, each in the interval $[-1, 1]$, by the viewport transformation

$$\mathcal{W} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} (x+1) \cdot (|U|-1)/2 \\ (y+1) \cdot (|V|-1)/2 \end{bmatrix} + \mathbf{1}. \tag{2.4.10}$$

The vertices does not necessarily lie on the discrete image coordinates, yet. In Open Graphics Library (OpenGL), this is achieved during rasterization were the vertex primitives are replaced by fragments that lie on the discrete positions defined by $U \times V$.

2.4.4 The OpenGL Rendering Pipeline

OpenGL is a widely used Application Programming Interface (API) for hardware accelerated rendering. OpenGL is constructed as a state machine, processing a stream of vertex data according to its current state. Over the past years the concept of the fixed function rendering pipeline in OpenGL evolved into a more flexible shader architecture, driven by the demand to harness the GPU for general purpose computing (GPGPU) and the need for more liberty in custom tailored rendering by replacing fixed function functionality with programmable shaders.

In OpenGL, scene objects are defined by describing the object's surface. A surface is build from primitives, which are typically triangles, and a triangle is formed by defining three vertices. The triangles can be processed independently, an essential condition for parallel data processing. Figure 2.8 shows the OpenGL rendering pipeline with the transformations at the traditional places. Some of the newer functionality of OpenGL has been added to the overview, allowing the relocation of some transformations. The vertices are defined in object coordinates O, and are first processed by the *vertex shader*. Before clipping, the vertices have to be transformed by the viewing transformation $\mathcal{Q}$ (Eq. 2.4.9), yielding camera coordinates. Then, the perspective $\mathcal{P}$ or orthographic projection matrix $\mathcal{O}$ is applied, yielding clipping coordinates. The OpenGL camera model and its projections are

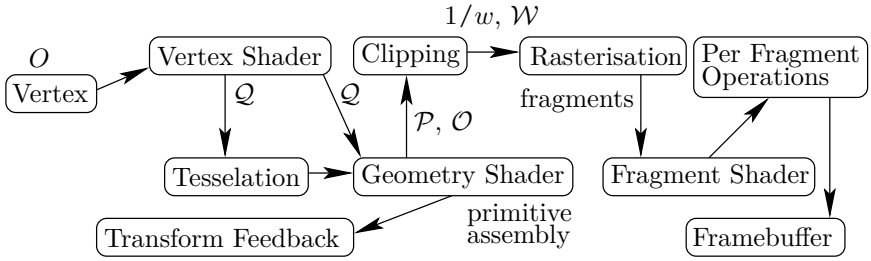

Figure 2.8. Overview of the *OpenGL* rendering pipeline. Object coordinates O, viewing transformation $\mathcal{Q}$, projections $\mathcal{P}$ and $\mathcal{O}$ respectively, perspective division $1/w$, and viewport transform $\mathcal{W}$.

discussed in detail in Section 2.4.6.

Traditionally, both transformations are applied in the vertex shader, as well as lighting, which is computed in eye coordinates. After the vertex shader, the primitives are assembled from the set of single vertices. By applying a *geometry shader*, the processing of primitives can be customised, allowing to create and insert new primitives into the rendering pipeline or modify existing ones. Next, the primitives are clipped against the viewing frustum, followed by the perspective division, yielding NDCs in the cubic interval of $[-1, 1]$. The primitives are transformed into window coordinates by the viewport transformation $\mathcal{W}$, followed by the *rasterization*. Rasterization produces a *fragment* for every pixel covered by the primitive, therefore a fragment can be considered a potential pixel. The fragments are processed independently, by the *fragment shader*. The fragment shader allows to modify the colour of each fragment. At this stage, textures may be applied, as well. Before the fragments that pass the fragment shader are written to the *frame buffer*, a visibility test can be applied, based on z-buffering. Visibility is computed by comparing the depth value of the fragment, by default writing the fragment with the smallest distance to the camera centre into the frame buffer. The next section gives a brief introduction to the programming model of graphics hardware and possible applications, other than rendering (GPGPU).

2.4.5 Parallel Hardware: GPU

Graphic hardware operates on the stream processing model, i.e. a set of instructions, called a *kernel*, is applied to a large set of homogenuously structured data elements. Parallelization is achieved by independent processing of the elements at the same time, simultaneous applying a Single Instruction to Multiple Data (SIMD) elements.

Before geometry shader were introduced, graphic hardware had dedicated processors for the processing of vertex shader and fragment shader. Both were based on the stream processing model, applying instructions to vertex data and fragment data respectively. The increasing parallel processing capabilities lead to the idea to provide a general access to these computational resources, leading to the C for Graphics (Cg) programming language (see Mark et al. [MGAK03]). By this time the OpenGL Shading Language (GLSL) was released, first as an OpenGL extension that later became part of the API, as of OpenGL 2.0 (see Rost [Ros05]).

The idea of utilising the computational resources of the graphics hardware for GPGPUs lead to several implementations, amongst others for a stream programming language (see Buck et al. [BFH+04]). For an early survey of GPGPUs, GPU programming languages, the GPU programming model, and applications, see Owens et al. [OLG+07].

Today, graphics hardware has evolved from different specialised processors to a "scalable array of multithreaded streaming multiprocessors"[1], where each multiprocessor executes a large amount of threads concurrently and is able to compute the different shader types. Manufacturer of graphic hardware released their own programming models with high and low level APIs, allowing to fully exploit the capabilities of the hardware, e.g. Compute Unified Device Architecture (CUDA) (see Nickolls et al. [NBGS08]).

This lead to a wide range of applications for GPGPUs, including computer vision (see, e.g. Bartczak et al. [BJK08]), digital image and video processing, bioinformatics, or computational finance. If the application and the data structures map well to the rendering pipeline, it can be beneficial to use GLSL, instead of a programming model that is close to the hardware, e.g. CUDA, relieving the programmer from the necessity to fine-tune the implemented algorithm to the hardware (e.g. see Jordt-Sedlazeck et al. [JSJK13]). In the next section the different projections and camera models supported by OpenGL are introduced.

[1] *CUDA* C Programming Guide, v5.5, NVIDIA Corporation, July 2013

2.4.6 Cameras

In this work, OpenGL is used for the rendering of computer generated content for full parallax displays and also for the acceleration of interpolated content. In the following, different camera projections are introduced, starting with the viewing frustum based cameras found in OpenGL that are described by projection matrices. Afterwards, a general ray based camera model is described that is often used in ray tracers and allows the definition of custom cameras for rendering. Ray tracers are often used to produce state of the art image content that exceed the capabilities of real-time orientated render frameworks. Finally, an equidistant fisheye projection is described, which is used to model the projection of one lens system of a lens array, as described in Section 2.3.

Pinhole Camera

According to Lindberg [Lin68], the first written correct analysis of the principles of a pinhole camera can be traced back to Al-Haitham, whose Arabic work has been translated into Latin in the thirteenth century. It explains the process of image formation caused by small apertures, as in pinhole cameras. The left side of Figure 2.9 shows a camera obscura, a

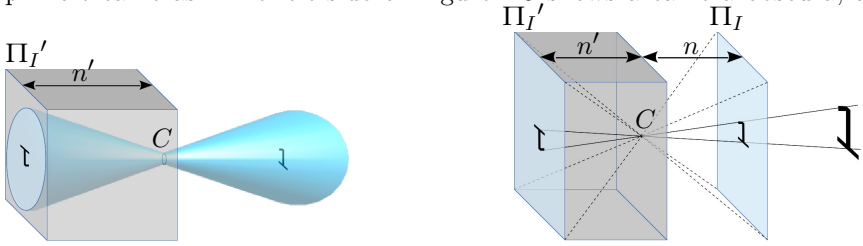

Figure 2.9. Sketch of a pinhole camera (left) and transition to an *OpenGL* frustum based camera (right) with an image plane Π_I located in front of the camera centre.

closed box with a hole, projecting a point reflected image onto its back, the image plane $\Pi_I{}'$. An ideal pinhole camera has a hole of infinitesimal size, therefore all light rays pass through a geometric point, the camera centre or optical centre, that defines the origin of the camera's coordinate system C. The distance between the camera centre C and the image plane $\Pi_I{}'$ is called the focal length n'.

Taking only the light rays into account that enter the camera housing through the pinhole, the image plane could be placed in front of the camera (Π_I, Fig. 2.9, right) with the same distance to the camera centre n, resulting in an upright image of the world that is not point reflected. The image formation of an ideal pinhole camera can be described by a perspective projection that is commonly used in computer graphics.

The OpenGL Camera Model

Although the projection matrix can be set with arbitrary values, in the following, two common ways to set the projection matrix are discussed, which result in a perspective and an orthographic projection.

Perspective Projection

During clipping, all vertices that lie outside the viewing frustum defined by the projection matrix are discarded from further processing. The viewing frustum of a perspective camera is depicted in Figure 2.10 (right). The projection matrix can be set by defining a rectangle I_n on the Near Clipping Plane (NCP)

$$\Pi_I := \left\{ (x, y, z)^T_{\in} \mathbb{R}^3 : z = n \right\};\ n_{\in} \mathbb{R}_{<0} \tag{2.4.11}$$

via the outer positions on the X-axis ($l_{\dashv}$, $r_{\dashv}$) and Y-axis ($b_{\dashv}$, $t_{\dashv}$) by

$$\begin{aligned} I_n := \left\{ (x, y, z)^T_{\in} \mathbb{R}^3 : x_{\in} [l_{\dashv}, r_{\dashv}], y_{\in} [b_{\dashv}, t_{\dashv}], z = n \right\}; \\ r_{\dashv}, t_{\dashv \in} \mathbb{R};\ l_{\dashv \in} \mathbb{R}_{<r_{\dashv}};\ b_{\dashv \in} \mathbb{R}_{<t_{\dashv}};\ n_{\in} \mathbb{R}_{<0} \end{aligned} \tag{2.4.12}$$

and limiting the frustum to the Far Clipping Plane (FCP), intersecting the Z-axis at $f_{\in} \mathbb{R}_{<n}$, parallel to the image plane Π_I. The frustum is defined in camera coordinates and the perspective projection matrix

$$\mathcal{P} = \begin{pmatrix} \frac{2n}{l_{\dashv} - r_{\dashv}} & 0 & \frac{r_{\dashv} + l_{\dashv}}{r_{\dashv} - l_{\dashv}} & 0 \\ 0 & \frac{2n}{b_{\dashv} - t_{\dashv}} & \frac{t_{\dashv} + b_{\dashv}}{t_{\dashv} - b_{\dashv}} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{2.4.13}$$

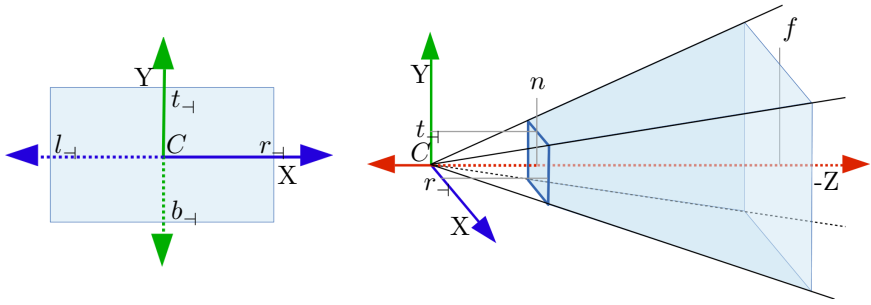

Figure 2.10. Definition of the near clipping plane located at n on the Z-axis (left) and perspective viewing frustum (right). Consistent with *OpenGL*, the camera is located at the origin, pointing in the negative direction of the Z-axis.

is defined such, that it transforms the frustum with its corners $(r_\dashv, t_\dashv, f)^T$ and $(l_\dashv, b_\dashv, n)^T$ from camera coordinates to the cubic interval of $[-1, 1]$ with its corners at $(1, 1, 1)^T$ and $(-1, -1, -1)^T$ in NDC, after applying the perspective division.

Orthographic Projection

In contrast to an ideal pinhole camera, were all light rays pass through a geometric point, the light rays of an orthographic projection are all parallel. Figure 2.11 shows an orthographic viewing frustum, defined in camera

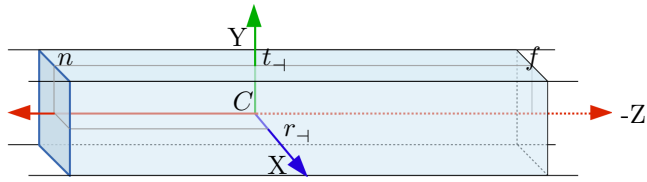

Figure 2.11. Orthographic viewing frustum. In contrast to the perspective projection the near clipping plane is not restricted to the negative Z-axis.

coordinates. The viewing frustum is defined via the same variables used in Eq. 2.4.11 and Eq. 2.4.12 with the exception that the restriction of positioning the NCP Π_I in Eq. 2.4.11 is lifted to arbitrary values $n \in \mathbb{R}$ on

the Z-axis. Again, the defined viewing frustum is transformed to the cubic interval of $[-1, 1]$ in NDC by the orthographic projection matrix

$$\mathcal{O} = \begin{pmatrix} \frac{2}{r_\dashv - l_\dashv} & 0 & 0 & \frac{r_\dashv + l_\dashv}{l_\dashv - r_\dashv} \\ 0 & \frac{2}{t_\dashv - b_\dashv} & 0 & \frac{t_\dashv + b_\dashv}{b_\dashv - t_\dashv} \\ 0 & 0 & \frac{2}{f-n} & \frac{f+n}{n-f} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.4.14}$$

leaving the last component of the homogeneous coordinates unaltered for the homogenisation, which is applied after clipping.

General Ray Based Cameras

For highest rendering quality, ray tracers are used. Ray tracer support rendering effects that are computational too demanding for real-time rendering or does not fit the data processing model used by hardware accelerated rendering. In contrast to OpenGL based rendering, were the geometry is transformed into the image plane, ray tracers invert the light path, casting rays through the image plane and intersecting those rays with the scene. This allows for custom camera models by defining a ray for each image pixel $_\in U \times V$

$$U \times V \to \mathbb{R}^3 \times \mathbb{R}^3 \tag{2.4.15}$$

by the ray origin and ray direction in camera coordinates. Hence, the ray origin is constant (usually $\mathbf{0}$) and the direction is normalised and points towards the related pixel on the image plane. The ray origin can then be used to modify the start of the ray along its direction, before the ray is later transformed into world or object coordinates. In the following, this is used to render images using an equidistant fisheye projection, the projection model of a single lens system of the lens array based display.

Equidistant Fisheye Projection

Figure 2.12 (left) shows a spherical coordinate system with azimuth angle θ, polar angle φ, and distance to the coordinate centre r. For an equidistant fisheye projection

$$r' = r \cdot \varphi = \varphi; \text{ with } r = 1 \tag{2.4.16}$$

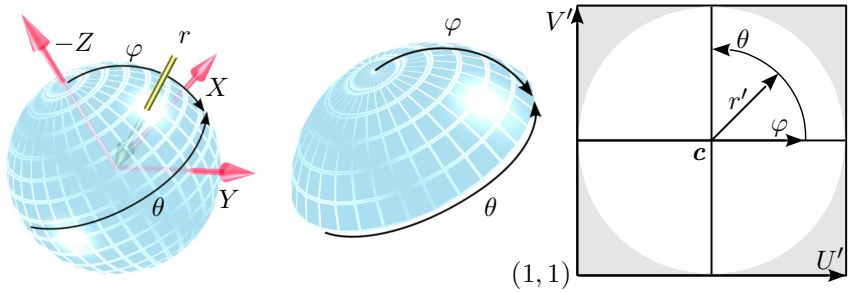

Figure 2.12. Spherical image representation. The left image shows the spherical coordinates (r, θ, φ), the centre image shows the part clipped to the ψ of the camera and the right image shows the mapping between spherical coordinates, projected onto the unit sphere ($r = 1$) and the image coordinate frame $U \times V$.

holds (see Schneider et al. [SSM09]). The right side of Figure 2.12 shows an equidistant fisheye image in the image coordinate frame $U \times V$. The 2D image coordinates are transformed into normalised image coordinates of the interval $[-1, 1] \times [-1, 1]$ with the inverse viewport transformation ($\mathcal{W}^{-1}$, cf. Eq. 2.4.10)

$$\mathcal{W}^{-1} : \begin{bmatrix} u \\ v \end{bmatrix} \mapsto \begin{bmatrix} 2 \cdot (u-1) \,/\, (|U|-1) \\ 2 \cdot (v-1) \,/\, (|V|-1) \end{bmatrix} - \mathbf{1}. \tag{2.4.17}$$

The bright circular area in Fig. 2.12 (right) represents the axially symmetric field of view ψ of the fisheye projection and is defined by

$$r' = \|\mathcal{W}^{-1}(u, v)\|_2; \text{ with } r' \in \mathbb{R} \overset{!}{\leqslant} 1, \tag{2.4.18}$$

thus restricting the maximum distance to the normalised coordinates origin to one. The image centre $\boldsymbol{c} \in \mathbb{Q}^2_{>0}$ is defined by

$$\boldsymbol{c} = \begin{bmatrix} (|U|-1)\,/2 \\ (|V|-1)\,/2 \end{bmatrix} + \mathbf{1} \tag{2.4.19}$$

and represents the optical axis of the fisheye projection, aligned with the Z-axis of the camera coordinate system. The 2D vector $\boldsymbol{n}^{2D}{}_{\in}\mathbb{R}^2$ is normalised

$$\boldsymbol{n}^{2D} = \begin{cases} \mathcal{W}^{-1}(u,v)\,/r'; & \text{if } r' \neq 0 \\ [0,0]^T; & \text{else,} \end{cases} \tag{2.4.20}$$

effectively fixing the azimuth angle θ on the unit sphere. The polar angle φ is assigned by a linear mapping of the radius r' to the Angle Of Field (AOF) $(0.5 \cdot \psi)$ of the unit sphere fisheye projection by

$$\varphi = r' \cdot \frac{\psi}{2}. \tag{2.4.21}$$

The normalised direction of a ray is defined by the origin of the camera coordinate system and a point on the unit sphere $\boldsymbol{\gamma}_{\in}\mathbb{R}^3$, corresponding to an image pixel $(u,v)_{\in}\, U \times V$ from the image coordinate frame, according to

$$\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{n}^{2D} \cdot sin(\varphi) \\ -cos(\varphi) \end{bmatrix}. \tag{2.4.22}$$

With the camera centre located in the origin, Equation 2.4.22 is used to define a ray based equidistant fisheye camera, allowing to render integral images for the lens array based display introduced in Section 2.3.

For the projection of a 3D point $\boldsymbol{p}_{\in}\mathbb{R}^3$ from the camera coordinate system to the image plane in NDCs, its normalised direction $\boldsymbol{p}_N = \boldsymbol{p}/\|\boldsymbol{p}\|_2$ is used to obtain the polar angle φ by its Z-axis component

$$\varphi = cos^{-1}\left(-\boldsymbol{p}_{N_Z}\right). \tag{2.4.23}$$

The parallel projection of $\boldsymbol{p}_N$ along the Z-axis into the XY-image-plane lies on a circle with radius $sin(\varphi)$. The normalised 2D vector $\mathbf{n}^{2D}$ is retrieved by

$$\boldsymbol{n}^{2D} = \begin{cases} \boldsymbol{p}_{N_{XY}}/\,sin(\varphi); & \text{if } \varphi' \neq 0 \\ [0,0]^T; & \text{else.} \end{cases} \tag{2.4.24}$$

The XY components of the NDCs are derived by the linear mapping to the AOF $\psi/2$ by

$$NDC_{XY} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \boldsymbol{n}^{2D} \cdot \frac{\varphi \cdot 2}{\psi}. \tag{2.4.25}$$

In OpenGL, vertex shaders can be utilised to transform the geometry of a scene via its vertices. Hence, is is possible to use Equation 2.4.25 to transform the scene in accordance to an equidistant fisheye projection to the unit cube of $[-1, 1]$, which will be discussed in detail in Section A.3 of the appendix. For proper visibility tests, clipping, and a beneficial mapping of the normalised depth values to scene depth, the Z-axis component of point $\boldsymbol{p}$ has to be transformed into the interval of $[-1, 1]$, as well. This is achieved by a linear mapping of the ray length of the desired interval in camera coordinates to the interval of the Z-axis component in NDCs by

$$NDC_Z : \begin{bmatrix} x \\ y \\ z \end{bmatrix} \mapsto \frac{2}{f-n} \cdot \frac{\boldsymbol{p}_Z}{|\boldsymbol{p}_Z|} \cdot \|\boldsymbol{p}\|_2 + \frac{f+n}{n-f}. \tag{2.4.26}$$

The viewing volume of the ray based camera is usually symmetric around the camera centre, which is included (cf. Figure 2.6). Hence, for rendering with OpenGL, the viewing frustum has to be rendered in two rendering passes, see Section 2.6.3 and Appendix A for a thorough explanation. Especially the length of the ray is ambiguous, because it can be both behind and in front of the camera centre. Therefore, the signum of the Z-axis component has to be preserved.

2.4.7 Relation of Different Representations

The light field of a multi-view display can be stored as 2D perspective images or as 2D orthographic images, as well. Relating the light field of a multi-view display based on lens arrays to the plenoptic function (cf. Equation 2.3.6) the position $\mathbf{V}$ relates to the projection centre of the lens system. With Equation 2.3.4, the incidence angle (φ, θ) of a light ray, passing through a projection centre, relates to the view direction of the display. Hence, Equation 2.3.1 can be written as

$$P : \mathrm{Y} \times \mathbb{R} \times \mathbb{R} \times W \mapsto \mathbb{N}_{\leqslant 255} \tag{2.4.27}$$

$$P\left((\varphi, \theta), \lambda, t, (\boldsymbol{V}_X, \boldsymbol{V}_Y, \boldsymbol{V}_Z)\right). \tag{2.4.28}$$

The lens systems of the display all lie in a plane, therefore the world coordinate system W can be aligned such that all projection centres are in $\boldsymbol{V}_Z = 0$, hence reducing the number of variant parameters of Equation 2.4.28

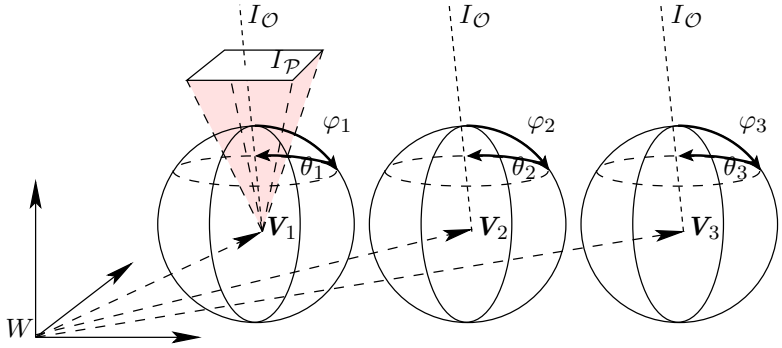

Figure 2.13. Plenoptic representation of a light field encoded in perspective $I_{\mathcal{P}}$ and orthographic $I_{\mathcal{O}}$ 2D images.

by one.

For a single perspective image $I_{\mathcal{P}}$, cf. Figure 2.13, the plenoptic function is fixed at the position $\mathbf{V}$ and has a varying incidence angle (φ, θ) of the stored light rays. The complete light field, stored as a set of perspective images is therefore the union over all perspective images, one per lens system R_v, constructed by

$$LF_{\mathcal{P}} = \bigcup_{p \in B(R_v)} P_p\left((\cdot,\cdot), \cdot, \cdot, (p_x, p_y, 0)\right). \tag{2.4.29}$$

In contrast, an orthographic image $I_{\mathcal{P}}$ stores a single viewing direction. Hence, the plenoptic function has a fixed incidence angle (φ, θ) per orthographic image but varies in the rays position $\mathbf{V}$. Analogous, the complete light field is included in the union over all views T, as

$$LF_{\mathcal{O}} = \bigcup_{o \in A(T)} P_o\left((\varphi_o, \theta_o), \cdot, \cdot, (\cdot,\cdot, 0)\right). \tag{2.4.30}$$

Conversion between Perspective and Orthographic Representations

A light field in a perspective representation can be converted into an orthographic representation and vice versa. The drawback is that the conversion is a tedious process, due to the data access pattern. In order to gain

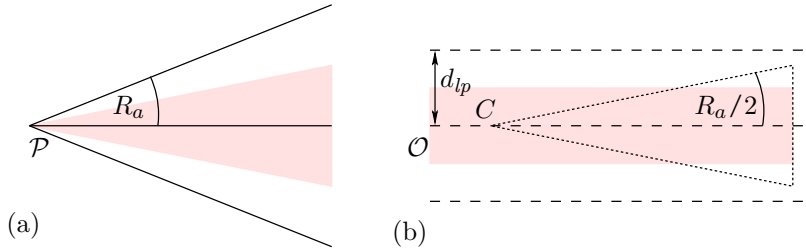

Figure 2.14. Perspective sampling (a) and orthographic sampling (b) of neighbouring light rays.

the perspective image of a single lens system P_p from the orthographic representation, the whole set of orthographic images have to be accessed, picking every view direction at that position out of the domain of all viewing directions

$$P_p = \bigcup_{o \in A(T)} P_o\left((\varphi_o, \theta_o), \cdot, \cdot, (x_p, y_p, 0)\right). \tag{2.4.31}$$

Converting images in the opposite direction poses a similar problem. For retrieving an orthographic image for a single view direction P_o from the perspective representation, the whole set of perspective images have to be accessed, collecting the desired view direction from every lens system

$$P_o = \bigcup_{p \in B(R_v)} P_p\left((\varphi_o, \theta_o), \cdot, \cdot, (x_p, y_p, 0)\right). \tag{2.4.32}$$

As long as the complete light field fits into memory, this is no drawback, but as soon as the light field exceeds the available memory, the complete orthographic image set has to be read at least once from the hard disk drive. If the light field is much larger as the available memory on the workstation, the complete light field has to be read multiple times from the hard disk drive.

Implementation

The Figures 2.14 show the footprint of one pixel under perspective projection (a) and under orthographic projection (b) as a transparent area. Based on the angle between neighbouring light rays R_a, the distance between neighbouring light rays increase with an increasing distance to the camera

centre under perspective projection. Hence, a pixel of a perspective image has the frustum of a pyramid.

In contrast, under orthographic projection the distance between neighbouring light rays is the constant value of the distance between neighbouring lens systems d_{lp}, regardless of the distance to the camera centre. Therefore, the footprint of a pixel under orthographic projection is a parallelepiped.

When converting images from the perspective representation into the orthographic representation, and vice versa, the different footprints of the pixels are ignored. The projection model for the lens systems of the display is that of an equidistant fisheye projection. Hence, rendering via ray tracers is done under that projection model, resulting in pixels that have the frustum of a pyramid. When the rendered images are transferred into an orthographic representation, the direction of the light ray in the centre of the frustum corresponds to the orthographic projection model, but the footprint of a pixel is the frustum of a pyramid (stippled area in Figure 2.14 (b)). The benefit is that orthographic images, which have these properties, can be used for storage of the light field and interpolation, without introducing additional aliasing due to a different footprint of a pixel and resampling of the image.

2.5 Comparing Images

In this section the metrics for measuring the fidelity of images are introduced. Over the years the PSNR has been frequently criticised (cf. Lambrecht and Farrell [vdBLF96], Wang et al. [WBSS04], and Winkler and Mohandas [WM08]) as an unfit metric for judging the perceived visual quality of an image. Nevertheless, PSNR is widely used, and as a consequence, researchers are familiar with its interpretation. For this reason it was chosen to use the Average Distance (AD) and the PSNR for measuring the fidelity of images. The AD is defined pixel wise between the reference image I and the target image I' as

$$AD\left(I, I'\right) = \frac{\sum\limits_{v \in V} \sum\limits_{u \in U} \sum\limits_{c \in c} \left| I\left(u, v, c\right) - I'\left(u, v, c\right) \right|}{|U| \cdot |V| \cdot |c|}. \tag{2.5.1}$$

The PSNR is defined via the Mean Squared Error (MSE). The MSE is defined as the sum over the squared difference of all pixels between the

reference image I and the target image I'

$$MSE\left(I, I'\right) = \frac{1}{|U| \cdot |V|} \sum_{v \in V} \sum_{u \in U} \left(\frac{\sum_{c \in C} \left| I\left(u, v, c\right) - I'\left(u, v, c\right) \right|}{|c|} \right)^2, \quad (2.5.2)$$

averaging the different channels c of the images. The PSNR is defined as

$$PSNR\left(I, I'\right) = 10 \cdot log \frac{255^2}{MSE\left(I, I'\right)}, \quad (2.5.3)$$

assuming the data consists of images with 8-Bit per channel.

2.6 Rendering for Full Parallax Displays

Untracked autostereoscopic displays work by offering the light rays that intersect the display for all viewpoints within the defined viewing zone of the display. For displays based on lens arrays, the image information is represented by 2D images behind each lens system (see Figure 2.5). The lens system itself can be thought of as a tiny projector, each with a centre of projection. The projected light rays of all lens systems of the display can be described as a light field. In the following, different methods to render computer generated light fields for autostereoscopic multi-view displays are introduced.

2.6.1 Content Rendering with Ray Tracers

There are several modelling tools available that allow the rendering of the modelled scene via ray tracers. Most ray tracers offer a plug-in interface for the implementation of custom cameras, therefore allowing the rendering of the elemental images, as described in Section 2.4.6.

However, the drawback is that models can not be easily transferred between different modelling tools, and often effects or objects of the scene are lost or rendered defective, sometimes even after transition to the next version of the same modelling tool. This constrains the content creation process and either limits the support to a small number of modelling tools or requires to constantly support a large number of plug-ins for different modelling tools.

Ray tracers have many applications and can be used to render models of highest complexity (see Wald et al. [WDS04]). Nevertheless, ray tracing is computational expensive, hence there have been many approaches to reduce the rendering time by accelerating ray tracers. Weghorst et al. [WHG84] introduced the idea to use z-buffering to determine visibility, which substantially accelerated the initial ray cast and can efficiently utilise the GPU. Carr et al. [CHH02] and Purcell et al. [PBMH02] were the first to map ray tracing to the GPU, utilising programmable shading.

Dietrich et al. [DWBS03] introduced *OpenRT*, an API for ray tracing, inspired by OpenGL. The goal was to create a common interface that allows to port existing OpenGL applications to this API, support ray tracing of existing 3D models, and deliver ray tracing support for scene graph libraries.

In addition to GPUs, several other hardware platforms were successfully utilised to accelerate ray tracing, e.g. FPGAs (Schmittler et al. [SWW+04]) or Cell processors (Benthin et al. [BWSF06]), ultimately leading to an integration of the parallel processing capabilities of GPUs and multi core CPUs (Georgiev et al. [GRHS08]) in order to further accelerate ray tracing.

Besides accelerating the computational expensive ray tracing by parallelization, there are approaches to cache and reuse computations. Dietrich et al. [DSS06], [DS07] achieved a remarkable acceleration in real-time ray tracing by sample caching and reuse of shading computations, exploiting frame-to-frame coherence. Similar to texture MIP map levels they cached computation results on different levels of the hierarchical space partition for reuse. Caching is also used for accelerating the illumination (Georgiev et al. [GKPS12]) and computation of the visibility, especially in complex scenes (Popov et al. [PGSD13]).

Finally, recent work of Davidovič et al. [DEG+12] aims towards a unified rendering concept that unifies ray casting and 2D rasterization with the goal to join the benefits of both and accelerate costly computations like global effects.

Ray tracers are commonly used for the rendering of light fields. Heide et al. [HWRH13] proposed to render light fields for compressive displays by an iterative approach of alternating taking plenoptic samples, followed by an optimisation in order to find the best samples for the next iteration, minimising the residual. Depending on the scene, they reduced the rendered light rays to 1.62% - 11.1%. For a light field of $1.1 \cdot 10^9$ rays, they reported a rendering time of about 99 hours with an additional hour for the optimisation.

2.6.2 Point Based Rendering Methods

Traditionally, hierarchical volumetric data structures like kd-trees have been managed and processed on the CPU. The reasons are the potential large memory requirements of dense 3D data and the conditional processing of the hierarchical data structure, which imposes a challenge to the Single Instruction Multiple Data (SIMD) execution model of GPUs.

For high quality rendering of volumetric data, splatting methods have been introduced (see Westover [Wes90], Zwicker et al. [ZPvBG01]).

Since then, processing and rendering of hierarchical data structures has been shifted to the GPU, in order to exploit the parallel processing capabilities (see Zhou et al. [ZHWG08], Keller et al. [KCK09] and [KLL+13]).

2.6.3 Rendering of Elemental Images with OpenGL

One important issue is the rendering speed for efficient view interpolation. Halle and Kropp [HK97] used the technically mature rendering of OpenGL for the rendering of perspective images for full parallax displays. This way they took advantage of the parallelization on the graphics hardware with a minimum effort and full compatibility with the graphics toolkits that are based on OpenGL.

Based on the work of Halle and Kropp, Holzbach and Chen [HC02] developed an algorithm that avoids rendering artefacts introduced by the clipping of polygons at the near clipping plane and handles degenerate cases, allowing for a commercial application.

Balogh et al. [BKB07] uses an OpenGL wrapper between the application and their display to render computer graphics content for their display, allowing to display content rendered with OpenGL. Annen et al. [AMZ+06] used a distributed rendering system to render images for autostereoscopic displays at interactive rates. They implemented several distributed rendering algorithms for Chromium [HHN+02] and rendered images for front and rear projection autostereoscopic displays.

Figure 2.15 shows a sketch of the 2D viewing frustum of a single lens system, with its centre located in the display plane. Inside the viewing frustum are two objects, a circle in front of the display plane and a square behind the display plane. The light ray of one pixel of the elemental image is drawn as a solid line before the first intersection with an object of the scene, and stippled afterwards. The first intersection of the light ray with

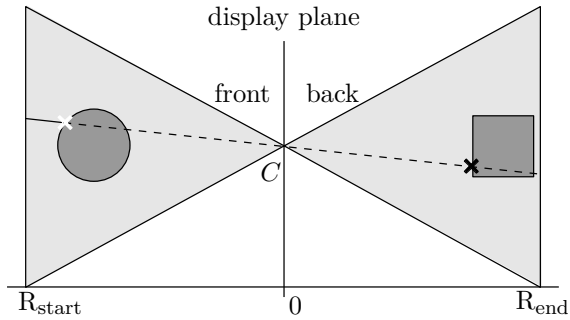

Figure 2.15. Viewing frustum of an elemental image of a full parallax display (sketch in 2D).

each object is indicated by a solid cross.

The viewing frustum in Figure 2.15 can not be defined by an OpenGL projection matrix. The NCP has to have a positive distance to the camera centre to avoid a division by zero, due to the perspective division. Halle and Kropp [HK97] proposed to render the objects behind the display plane and in front of the display plane separately. In order to render objects that lie in the display plane, they proposed to shift the camera by a small amount and locate the NCP at the display plane. The part of the scene that lies behind the display plane could be rendered with OpenGL without any adjustments. For the part in front of the display the depth test has to be inverted, and Halle and Kropp had to adapted the reflection model and shading, as well. The final elemental image was retrieved by merging the contents of the scene behind the display plane, and the content in front of the display plane, which is point reflected at the camera centre.

A more detailed introduction to the rendering of perspective elemental images for full parallax autostereoscopic displays can be found in Appendix A, as well as an implementation of a fisheye projection in OpenGL using vertex shader.

2.6.4 Rendering of Orthographic Images with OpenGL

The rendering of orthographic images for a multi-view display with OpenGL is straight forward and possible without the modifications of lighting or

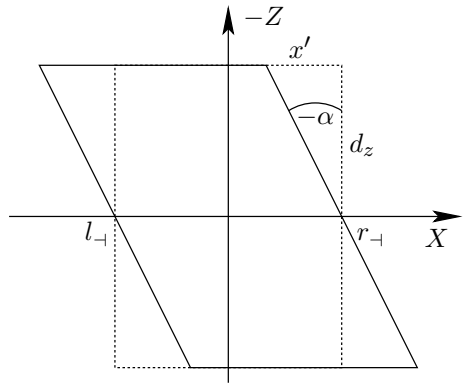

Figure 2.16. Orthographic rendering of the view directions of a multi-view display with *OpenGL*.

the visibility test, needed by the pseudoscopic rendering of the perspective images. Also, the rendering of the foreground and background geometry can be done in one single rendering pass, without the need to take the point reflection into account or combine the rendering results afterwards.

Figure 2.16 shows a 2D orthographic viewing frustum (stippled line), with the virtual display plane located in the XY-plane. Relating to the viewing rays of the multi-view display, the stippled viewing frustum renders the light rays under the incidence angle $(0, 0)$

$$P_o\left((0,0),\cdot,\cdot,(\cdot,\cdot,0)\right), \tag{2.6.1}$$

for all lens systems of the display. In order to render all light rays under the incidence angle $\alpha, \beta_{\in}[-\psi/2, \psi/2]$ the orthogonal viewing frustum has to be transformed by an affine transformation. The solid drawn frustum of Figure 2.16 is sheared for rendering of the negative incidence angle α. The resulting translation x' along the X-axis depends on the z-coordinate of the vertices d_z and the view direction α

$$x' = d_z \cdot tan\left(\alpha\right). \tag{2.6.2}$$

The offset x' has to be transformed to NDCs and is applied to the ortho-

graphic projection matrix

$$\mathcal{O} = \begin{pmatrix} \frac{2}{r_\dashv - l_\dashv} & 0 & \frac{2 \cdot \tan(\alpha)}{r_\dashv - l_\dashv} & \frac{r_\dashv + l_\dashv}{l_\dashv - r_\dashv} \\ 0 & \frac{2}{t_\dashv - b_\dashv} & \frac{2 \cdot \tan(\beta)}{t_\dashv - b_\dashv} & \frac{t_\dashv + b_\dashv}{b_\dashv - t_\dashv} \\ 0 & 0 & \frac{2}{f - n} & \frac{f+n}{n-f} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.6.3}$$

with an analogous transformation on the Y-axis.

2.7 DIBR for Multi-View Displays

The last section introduced rendering methods for full parallax displays. Although real-time oriented rendering APIs like OpenGL are well suited to render light fields for static content displays in a minimum of time, the next generation of high resolution video multi-view displays offer such a vast amount of views, that real-time rendering and transfer of the image data to the device is not feasible. Moreover, some applications depend on special modelling tools and proprietary ray tracers for image rendering.

For such applications, DIBR is used to interpolate the full data set from a sparse sampled set of colour and depth input images. After an introduction to DIBR, the application of DIBR to multi-view displays are discussed, which lead to special requirements and constraints.

2.7.1 Related Work

The representation of light fields (see Section 2.3.1) and IBR are closely related. Under the assumption that a ray of light has a constant radiance, the 5D light field that parametrises the position and angle of a static and constantly illuminated scene can be reduced to a 4D function. This way it is not possible to describe occlusions or transparency but it allows to predict the direct luminance intensity without requiring any knowledge of the geometry. According to Levoy [Lev06], the idea to reduce the light field to a 4D function originates from Moon and Spencer [MS81]. Since then, 4D light fields have been used in several applications (e.g. Levin [Lev71], Ashdown [Ash93]).

Levoy and Hanrahan [LH96] introduced the two plane parametrisation of the 4D function and the idea of rendering new viewpoints out of this

representation. Their parametrisation is well suited for the fast rendering of viewpoints and does not depend on the geometry of the scene. The ideas introduced by Levoy and Hanrahan [LH96] led to a large number of fast light field renderers.

Classification

Shum and Kang [SK99] classified Image Based Rendering (IBR) methods into rendering without geometry, rendering with implicit geometry, and rendering with explicit geometry.

Buehler et al. [BBM+01] systematically classified IBR algorithms further by a number of desirable goals. The classification covered the use of geometric information, consistency of epipoles, correct surface sampling, unstructured input positions, consistency of the interpolation of the same ray, small motion consistency, a minimal angular deviation of the input rays, and real-time capability.

Similar to Shum and Kang the light field rendering techniques are categorised into rendering that is solely based on the direction of light rays, techniques that use an approximate geometry for rendering, and rendering that is supported by a full geometry. However, these categories can not be utilised to clearly separate all light field renderer, as there are some renderer that support an optional depth correction, effectively allowing the algorithm to be placed in several categories (e.g. Gortler et al. [GGSC96] or Buehler et al. [BBM+01]).

Purely Directional Light Field Rendering

Light field rendering without geometry is typically based on a set of light rays of known position and direction, a representation that is tailored to suit the intended application, and a constant depth assumption for rendering, in order to reduce interpolation artefacts. A novel view is rendered from the light field by finding the best match for each viewing ray of the novel view in the set of available light rays.

Levoy and Hanrahan [LH96] introduced the two plane parametrisation for the light field, which is well suited for the fast rendering of novel views with full parallax. Especially the rendering is essentially a projective texture mapping, which is supported in hardware. The geometry is modelled by a

constant depth assumption, hence for light fields of single objects the depth is usually set to the object centre to reduce rendering artefacts.

Shum and He [SH99] introduced an efficient 3D parametrisation for light fields, which allows the rendering of novel views with a horizontal parallax. Shum and He used a slit camera, moved along concentric circles, to record 3D light fields, which are parametrised by the vertical FOV of the camera, the radius of the circle, and the rotation angle, yielding concentric mosaics for the different circle radii.

Szeliski and Shum [SS97] introduced a method to construct panoramic images from estimated rotation matrices and focal length. In order to reduce the accumulated registration error they introduced a global alignment based on block adjustment, using feature-based point correspondences [SS98]. A local alignment was used to compensate small camera translations, which would otherwise lead to ghosting artefacts.

A drawback of rendering without geometry is that it requires a lot of input samples to reduce ghosting due to an unmodelled scene geometry and is in general not well suited for scenes with large depth variations. Depending on the depth and the scene, the rendering error increases with an increasing baseline between the interpolated image and the input images. This characteristic contradicts the strategy to minimise the number of input images, leading to increasing ghosting artefacts at depth discontinuities with a decreasing number of input images.

Approximate Geometry Based Light Field Rendering

This drawback can be remedied by incorporating depth information or image correspondences into the interpolation process. In remote sensing, image warping (Wolberg and Boult [WB89]) has been introduced to correct distortions, allowing to register images with less blending artefacts (see Wolberg [Wol90]). The image transformation was estimated on the basis of correspondences.

When the image transformation is related to a projection model, image correspondences can be used to create novel views of the scene. Seitz and Dyer [SD96] created novel views located on the baseline between two parallel input views. In case the baseline is parallel to the image plane of the two views, the intermediate views are linear combinations of the input views. For input images that did not fulfil these requirements, Seitz and Dyer reprojected the input images onto a common plane.

Gortler et al. [GGSC96] also used the two plane parametrisation for rendering of novel viewpoints, but introduced a depth correction by a 3D model to reduce ghosting artefacts. For representation of the model they chose an octree, as described by Szeliski [Sze93], allowing for depth compensated interpolation based on shape approximations, as well as full geometric models. The idea of Gortler et al. allows to interpolate images based on very few input images without ghosting artefacts, given exact depth information.

Evers-Senne and Koch [ESK05] evaluated two different rendering methods using image and depth data from an uncalibrated multi-camera rig, without refining a globally consistent model of the scene. By refining one 3D mesh for the interpolated view, a local approximation of the scene was used to warp the neighbouring images. The other rendering approximated the scene geometry by regular quadrilaterals of adaptive size in the image plane. For refinement, properties of the quadrilateral in 3D space were used, subdividing the regular quadrilateral in a recursive refinement process in order to approximate the geometry of the scene.

Farre et al. [FWL+11] introduced an algorithm to render novel views for multi-view displays based on image domain warping that implicitly handles artefacts introduced by occlusions. Image correspondences were established by combining feature matching with optical flow.

Model Based Light Field Rendering

Chen and Williams [CW93] introduced a view interpolation technique based on image warping. Visibility was solved by ordered drawing from back to front and holes were filled by either linear interpolation or re-rendering of missing pixels. Due to the evaluation with ground truth disparity maps and the noise sensitive nature of the per-pixel-warping, the rendering technique is categorised as model based.

Debevec et al. [DTM96] introduced view-dependent texture mapping. The idea is to generate a basic model and to project the input images onto the model, weighted by the similarity to the virtual view. Later Debevec et al. [DYB98] used projective texture mapping to efficiently implement view-dependent texture mapping. A notable advance was the possibility to calculate visibility and therefore improve the quality in occluded regions.

Buehler et al. [BBM+01] evaluated IBR algorithms ([DYB98], [DTM96], [GGSC96], [HKP+99], [LH96], [PCD+97], [WAA+00]) on the basis of a set

of desirable goals and designed a novel rendering algorithm to meet these goals. By evaluating a sparse blending field for the input images they utilised the graphics hardware to interpolate a dense blending field and blend the input images, rendering free viewpoints at interactive rates.

Evers-Senne [ES08] introduced several IBR algorithms for different applications. Depending on the quality of the available depth maps, he proposed a scheme to select the appropriate rendering algorithm. If the depth maps were of insufficient quality he used a Plane-Sweep for view interpolation.

A voxel based approach to image interpolation has been introduced by Seitz and Dyer [SD97]. By projecting every voxel into every image, correspondences are established, yielding a consistent 3D model of the scene. This approach lead to the space carving algorithm (Kutulakos and Seitz [KS98]), which starts with a complete set of of voxels and iteratively removes non-photo-consistent voxels, following the visibility based sweep direction.

Todt et al. [TLRS+08] used spherical light field rendering to generate virtual views for pose estimation and object classification. The rendering was aided by a Photonic Mixer Device (PMD) depth camera, in order to obtain per pixel depth information.

Model and Illumination Based Light Field Rendering

Another approach to incorporate geometric support has been made by Lischinski and Rappoport [LR98]. They introduced the Layered Light Field (LLF), which incorporates parallel Layered Depth Images (LDIs), surface normals, diffuse shading, visibility of light sources, and the material properties at the surface point, allowing DIBR to handle non-diffuse synthetic scenes. This scene description contains the essential information for rendering and closes the gap between DIBR supported by a complete geometry, and full rendering systems like OpenGL and ray tracers, which render images from an abstract scene description. The LLF allows to render images of scenes that correctly handled view-dependent shading and allow for correct handling of glossy and reflective surfaces, outperforming ray tracers in rendering speed but maintaining comparable rendering quality at the same time.

An important task in order to provide content for multi-view displays is the conversion of 3D video formats. According to Merkle et al. [MMW10], 3D video formats can be divided into colour input, like two view video,

mixed resolution stereo, or multi-view video, and depth enhanced input, like video plus depth, multi-view video plus depth, and Layered Depth Video (LDV). Merkle et al. conclude that input data without depth enhancement could not be adapted to different display conditions in contrast to depth enhanced input data.

In horizontal parallax displays that are available commercially, usually LDV is utilised to render the different views out of a central view, a depth map, and an occlusion layer. The LDV format emerged from the LDI that was introduced by Shade et al. [SGHS98].

In general, most view interpolation algorithms can be applied to generate interpolated views for multi-view displays. However, when a view interpolation algorithm is used to generate images for a multi-view display there are special requirements that have to be met. These requirements arise from the production pipeline, the used multi-view display and the image interpolation process itself. In the following the inherited requirements for a view interpolation algorithm are introduced and the source of the requirement is described in detail.

The first set of requirements are inherited from the multi-view display itself and depend on its technical specification. In the remainder the focus will be placed on multi-view displays with a high spatial and angular resolution. This is justified with the technical development over the last years, which has shown that multi-view displays are increasing in both, the spatial and angular resolution. Therefore, the general requirements, later derived in Section 2.9, will have to take into account an increasing numbers of views and spatial resolution of current on the market multi-view displays, for the future development.

2.7.2 Data Rate of Multi-View Displays

In the past few years several full parallax displays for static content became available to the consumer market, which encode between 50,000 and up to 200,000 different views per pixel, based on holography and lens arrays (corporations: Zebra Imaging [HC02], REALEYES GmbH [JK11b]). Typical autostereoscopic displays for video range from 20 to 60 different views per pixel found in horizontal-parallax lenticular displays (Holliman et al. [HDFP11]), and up to about 300 different views per pixel in full 2D parallax lens arrays, deduced from Arai et al. [AOK+10]. A comprehensive survey of current multi-view displays was performed by

Holliman et al. [HDFP11] and Yaraş et al. [YKO10].

For an example, the production chain of a 3D poster of about 1 m^2 with static content is considered[2]. The content is typically produced by an artist of a company who creates a 3D model utilising a modelling tool, which often has an integrated ray tracer for rendering. The rendered images are then transferred to the production site of the display. The elemental images are projected onto the film during exposure and fixed by developing the film. After assembly of the display, the static light field of the scene is projected by the display. A full set of elemental images requires about 200 GB of memory[3], which can be reduced by lossless compression to about 60 GB. This imposes a high bandwidth requirement for transferring the image data and a high storage capacity for backing up the data.

With an increasing number of views, the requirements for storage and transmission of the high data rate of autostereoscopic video displays becomes a challenge. Arai et al. [AOK+10] and Mishina [Mis11] introduced a 3D TV system that allows the capturing of a scene and playback on an autostereoscopic display based on a lens array. For capturing they used a high resolution camera and for playback a high resolution projector of the same resolution. According to their specification, the capturing and display components had an uncompressed data rate of 100 MB per frame, resulting in 6.0 GB per second at the full frequency of 60 FPS.

Balogh et al. [BKB07] came to the conclusion that the data rate of future multi-view display technology will increase by 10^2 to 10^4 compared to current HDTV. Xu et al. [XPLL11] analysed the bandwidth requirements of a 3D holographic display. The bandwidth requirement of their display was about 1.3 GB per second and they solved the transmission challenge via a local network consisting of ten 1 Gbps channels. They came to the conclusion that the bandwidth requirement of autostereoscopic 3D displays will increase to the range of 12.5 GB to 125 GB per second, due to the increasing resolution of autostereoscopic displays. The authors proposed lossless compression or the transmission of 3D object data to deal with the increasing bandwidth requirements.

With an increasing spatial and angular resolution the consequence is a drastic increase in the resulting uncompressed data rate. The data rate d_{rate} is directly proportional to the number of views T, the resolution of a

[2] Corporation REALEYES GmbH

[3] 3 Byte per pixel, stored in elemental images of 512x512, with about 250,000 elemental images per m^2

single view R_v, the number of colour channels $|c|$ and the number of FPS Ξ

$$d_{rate} = \frac{T \cdot R_v \cdot |c| \cdot \Xi}{1024^2} \left[\frac{Mb}{s}\right]. \tag{2.7.1}$$

Given a hypothetical full parallax multi-view display with 40 x 40 views, each in High Definition Television (HDTV)[4], three colour channels with 8-Bit each, and 21 FPS the resulting uncompressed data rate will already be about 195 Gbytes per second.

This gives an example of the potential grow of the data rate for multi-view displays, that imposes three challenges. The rendering of the image data, its storage, and the data transfer to the display device.

Due to the regular alignment, the image data of a multi-view display in general has a lot of redundancy. The similarity of the images can be exploited in order to reduce the data size with compression techniques. Magnor et al. [MRG03] used a model-based coding to compress the large amount of image data needed for IBR and achieved high compression ratios of over 2000:1. Matusik and Pfister [MP04] implemented the full production chain for 3D TV from light field acquisition to display on an autostereoscopic display. For transmission they used a temporal encoding of the individual views with MPEG-2. Merkle et al. [MMSW06] utilised an Hx264/MPEG4-AVC codec to encode multi-view video streams exploiting similarities in the temporal and viewpoint domain.

However, data compression does not solve the challenge to render the full data set of the display, and although the data rate could be greatly reduced with compression techniques, in case of a video display a real-time capable decompression would have to be computed on the device. Due to this reasons another approach to reduce the amount of data has been chosen in this thesis.

Reducing the number of rendered images of the multi-view display will solve all three challenges by reducing the amount of data in the very beginning of the data processing. The drawback is that in order to obtain the full data set for display, the missing image data has to be interpolated. In the case of a static display, the interpolation should allow for a computational advantage when compared to rendering of the missing images with a ray tracer. For video displays, interpolation of the full data set has to be achieved in real-time. This leads to highly integrated and efficient hardware

[4]Depending on the format, HDTV is specified between 0.8 and 2.1 Mpixels.

solutions like GPUs, FPGAs, and Application-Specific Integrated Circuits (ASICs).

Algorithms and data flow of highly integrated parallel hardware often follow the SIMD principle, similar to vector computers and graphics hardware (cf. Section 2.4.5). The idea is to have the same instruction or set of instruction executed in parallel but on different data elements of the same kind. This has consequences for algorithms that are targeted towards this hardware. One is that branching of the program flow or a non-structured access on the input data usually lower the performance on this kind of hardware. Another is that the resources of the hardware often is divided among the number of instances of an instruction set[5], i.e. the instances executed in parallel. Hence, it is beneficial that the input data has as little impact as possible on the program flow and the interpolation sub-routine should allocate few resources.

To conclude, the amount of input data has to be reduced by a large factor to allow for rendering, storage, and data transmission to the display. A consequence is that the full data set has to be interpolated on the display device itself. In order to process the large amount of data, highly integrated parallel hardware could be utilised but at the cost of a structured input and constraints on the data flow, and the complexity of the interpolation algorithm.

2.8 An Introduction to Image Interpolation

Image and view interpolation methods allow to generate novel views from previously rendered or captures images. In the following the theoretical foundation for view interpolation is introduced. In particular, the special application of viewpoints that lie in a plane and share a common image plane is regarded.

2.8.1 Constraints Related to Multi-View Displays

The image data of lens array based displays lead to some constraints compared to the general case of view interpolation. The characteristic that all viewpoints lie in a plane has already been mentioned, in contrast to the general case of unrestricted viewpoint placement. Other constraints are

[5] *CUDA* C Programming Guide, v5.5, NVIDIA Corporation, July 2013

the regular placement of the views, the uniform projection model for all views, and, based on the distance to the display plane, a uniform sampling. Following the strategy of a reduced set of input images, another constraint is that the image interpolation algorithm should be able to cope with few input images, without introducing noticeable interpolation artefacts.

The following section extends the process of image formation of a single view (see Section 2.4) to a second view, and derives the relationship between the relative pose of two cameras, design of the scene, and the resulting correlation between both captured images.

2.8.2 Epipolar Geometry

The line connecting the two camera centres C_1 and C_2 is called the *baseline* $\mathfrak{b} := C_2 - C_1$. The intersection of the baseline with an image plane is called an *epipole*. In Figure 2.17 both epipoles happen to be in the active image area and project to image points C_2' and C_1', respectively. The two camera centres and any non-collinear 3D point span a plane, called an *epipolar plane* Π_e. All possible planes through both camera centres are called an *epipolar pencil*.

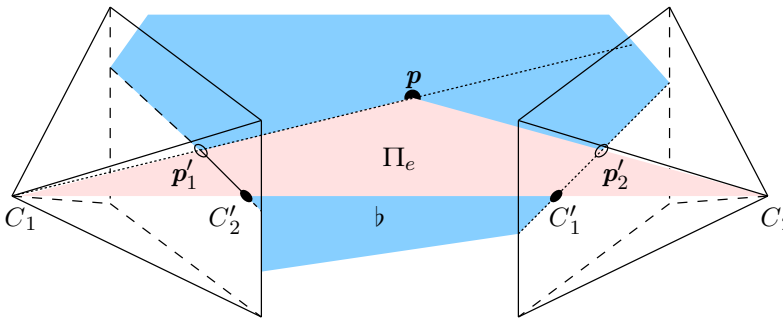

Figure 2.17. Epipolar geometry between two views.

Considering the fixed epipolar plane Π_e in Figure 2.17, defined by the camera centres C_1, C_2, and $\boldsymbol{p}$, the epipolar plane intersects with both image planes, forming the *epipolar lines* in the images (stippled line in view C_2, and dashed and solid line in view C_1). The projection of $\boldsymbol{p}$ onto the image planes is located on the corresponding epipolar line in each view. Moving $\boldsymbol{p}$

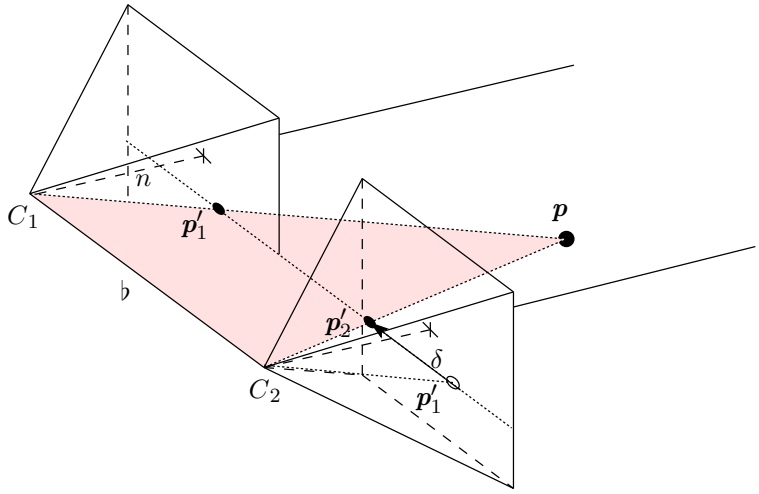

Figure 2.18. Epipolar geometry between two views of a lens-array based multi-view display.

along the direction defined by C_1 and $\boldsymbol{p}$ (fine dashed line), the location of its projection $\boldsymbol{p}'_1$ in view C_1 does not change. On the contrary, the projection $\boldsymbol{p}'_2$ in view C_2 moves along the epipolar line (fine dashed line), towards the epipole C'_1, when $\boldsymbol{p}$ is moved towards the camera centre C_1, and away from C'_1 when $\boldsymbol{p}$ is moved away from C_1.

Relating the general case to multi-view displays, the optical centres of all lens systems are coplanar, as well as all pixels on the image plane. Figure 2.18 shows this configuration for two views with the image plane of all views in distance n to the plane of projection centres. The optical axis of both views are parallel, intersecting the image in its centre. In contrast to the general case, the epipolar plane intersects only with the single image plane of all views and, as a result, the epipolar lines of two views lie on the same line in Euclidean space for any non-collinear 3D point $\boldsymbol{p}$. Another property of this special case is that the epipolar line is parallel to the baseline $\flat$. The disparity δ in Figure 2.18 is the 2D distance in image coordinates of the projected 3D point $\boldsymbol{p}'_1$ in view C_1 and $\boldsymbol{p}'_2$ in view C_2, measured in the image of view C_2. Due to the identical projection model for all views, the ratio of the disparity δ is split into fractions of the image coordinate system

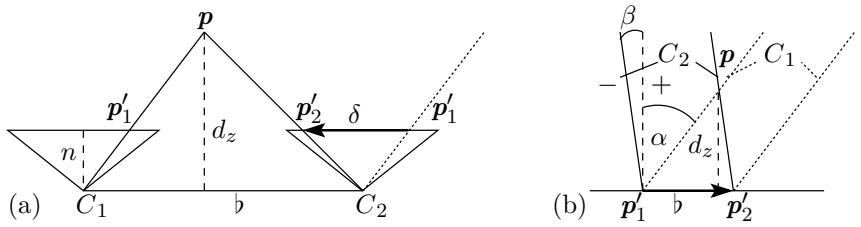

Figure 2.19. Relationship between depth d_z and disparity δ for two coplanar perspective views C_1 and C_2 (a) and two orthographic view directions (b).

$U \times V$, which directly depends on the difference in position of both views, with respect to the image coordinate frame $U \times V$. For the special case of coplanar camera centres and a single image plane, the relation between the distance of a 3D point to the baseline and the resulting disparity between two views can be easily derived.

2.8.3 Depth and Disparity

The relation between the distance of a point $\boldsymbol{p}$ to the baseline $\flat$ between the two perspective views C_1 and C_2 is depicted in Figure 2.19 (a). The camera's coordinate systems of both views are aligned, translated in the XY-plane of the camera's coordinate systems by the baseline $\flat$. Hence, the shortest distance of $\boldsymbol{p}$ to the baseline equals the Z-coordinate d_z of $\boldsymbol{p}$ in the camera's coordinate systems, in Figure 2.19. With the distance of the image plane to the camera centres n, the relation between depth and disparity for perspective views is therefore given by

$$\frac{d_z}{n} = \frac{\flat}{\delta}. \tag{2.8.1}$$

The relation between depth and disparity for orthographic views is depicted in Figure 2.19 (b). The view directions C_1 (stippled lines) and C_2 (solid lines) have the angles $\alpha, \beta \in [-\psi/2, \psi/2]$ to the optical axis (dashed line), with one side of the optical axis defined negative and the other positive. In contrast to the perspective views, the image plane of the orthographic views lies in the plane of the optical centres. The distance between the projection of point $\boldsymbol{p}$ in both orthographic views is related to the Z-coordinate d_z of

point $\boldsymbol{p}$ and the viewing direction by

$$\flat = d_z \cdot (\tan(\alpha) - \tan(\beta)). \tag{2.8.2}$$

Rendering of Depth in OpenGL

In ray tracers, depth is readily available. Often, the output format can be selected by the user, to choose between the Z-coordinate, the length d of the ray, or normalised depth. In OpenGL, the z-buffer can be accessed, allowing to obtain the result of the visibility test, which is the normalised depth z_n from the range $[0, 1]$. For perspective images, the depth in camera coordinates d_z can be obtained from the depth buffer by solving the concatenation of the Z-component of Equation 2.4.13, the perspective division, and the linear depth mapping to normalised depth, for d_z, yielding

$$d_z = \frac{f \cdot n}{(n - f) \cdot z_n + f}. \tag{2.8.3}$$

Solving Equation 2.8.1 for the disparity and replacing

$$n = \frac{1}{2 \cdot tan(\psi/2)}, \tag{2.8.4}$$

the 2D disparity is obtained component wise in width and height by

$$\delta(\nu, \xi) = \frac{\flat \cdot (\nu, \xi)^T}{d_z \cdot 2 \cdot tan(\psi/2)}, \tag{2.8.5}$$

scaled by the image size in width and height $(\nu, \xi)^T$ to obtain pixel dimensions.

Similar, the disparity can be obtained for orthographic images, solving the concatenation of Equation 2.4.14 and the linear depth mapping to normalised depth for d_z, resulting in

$$d_z = (f - n) \cdot z_n + n. \tag{2.8.6}$$

The mapping of all views of the display T to the angle of the view direction to the axes of the camera coordinates are defined by

$$H := \{(a, b)_{\in} \mathbb{R} \times \mathbb{R} \mid -\psi/2 \leqslant a \leqslant \psi/2, -\psi/2 \leqslant b \leqslant \psi/2\} \tag{2.8.7}$$

$$E : T \mapsto H. \tag{2.8.8}$$

The 2D disparity for the orthographic views C_1 and C_2, with the corresponding axis angles, $\alpha, \beta \in H$, is obtained component wise in width and height by dividing Equation 2.8.2, which is measured in camera coordinates, by the size of one pixel, yielding

$$\delta_{R_v} = d_z \cdot (\tan(\alpha) - \tan(\beta)) \cdot R_v \oslash \tilde{v} \; ; \; \tilde{v} := (r_{\dashv} - l_{\dashv}, t_{\dashv} - b_{\dashv})^T, \tag{2.8.9}$$

with the component wise vector division denoted as $\oslash$.

2.8.4 Warping of Images

Wolberg [Wol90] defines image warping as a geometric transformation, "an operation that redefines the spatial relationship between points in an image". According to Wolberg, image warping evolved from transformations on analogous images using optical systems, e.g. see Cutrona et al. [CLPP60]. Among the first applications of warping of digital images were terrain mapping. Since then, image warping found applications in remote sensing, computer vision and computer graphics. See Wolberg [Wol90] for a more complete overview of the history of image warping and its application.

The application of image warping in this work has the goal to transfer image content of a reference view C_1 into a novel target view C_2, with the aid of depth information of the reference view and the spatial relationship derived in the previous sections. In order to improve the warping result, filtering techniques can be utilised, which have been used for a long time in texture mapping, see Heckbert [Hec86] for a survey. In this application all views share the same image plane and the same projection (cf. Section 2.8.1), hence, a per pixel warping is used with an accumulation buffer technique or bilinear interpolation.

Given the disparity of the image of view C_1 to a corresponding image of view C_2, the image content can be transferred from C_1 to C_2 by

$$I_{C_2}(u + \lfloor D_{C_1}(u, v, 0) \rceil, v + \lfloor D_{C_1}(u, v, 1) \rceil, \cdot) = I_{C_1}(u, v, \cdot). \tag{2.8.10}$$

Equation 2.8.10 describes a pixel wise forward mapping of the image data, writing the image data to the nearest pixel position in the target view.

Different disparity values or quantisation effects may cause that more than one pixel of the reference image is transferred to the same pixel position

in the target view. In order to solve the visibility problem a disparity buffer is used, similar to the z-buffer method, as described by Hearn and Baker [HB97], comparing the disparity of the last warped pixel with the currently warped one. For perspective images, the disparity is inversely proportional to the distance of content to the optical centre (cf. Equation 2.8.5) and changes the sign when the content is in front of the display. This can be used to determine the content nearest to an observer and write to the target image accordingly. In contrast, for orthographic images the disparity is proportional to the distance of the content to the optical centre (cf. Equation 2.8.9), which has to be taken into account when visibility is determined.

Through the same mechanism, there might be pixels in the target image that get no colour value assigned. Holes of the size of one pixel, which originate from quantisation effects, and sampling errors, due to the discrete grid of image pixels, can be remedied by distributing the warped source pixel in the neighbourhood of the true warping position, based on the distance.

The disparity map $D_{i=1}$ of the source image is used to compute the 2D position in the target image, where the colour information is warped to, by

$$S : U \times V \mapsto \mathbb{R}^2. \tag{2.8.11}$$

$$S_i\left(a,b\right) = \left[\begin{array}{c} a + D_i\left(a,b,0\right) \\ b + D_i\left(a,b,1\right) \end{array}\right]. \tag{2.8.12}$$

Given a position $(u,v)_{\in}U \times V$ in the target image C_2, the warped colour information from the source image C_1 is distributed in the 3×3 neighbourhood of the target image. The corresponding source image coordinates $\overleftrightarrow{\mathbb{K}}^i_{u,v}$, which contribute to the target pixel (u,v), are defined by

$$\overleftrightarrow{\mathbb{K}}^i_{u,v} = \left\{ \left[\begin{array}{c} a \\ b \end{array}\right]_{\in}U \times V \middle| \left[\begin{array}{c} u' \\ v' \end{array}\right] = S_i\left(a,b\right); \left|\lfloor u' \rceil - u\right| \leqslant 1; \left|\lfloor v' \rceil - v\right| \leqslant 1 \right\}. \tag{2.8.13}$$

This approach leads to bleeding of the colour information. In order to reduce bleeding, the colour information is weighted by the distance to the true 2D warping position and the z-buffer test to stop the bleeding of colour information accross object boundaries. The distance of the true 2D warping position $c' = S_i\left(a,b\right)$ of a source pixel at position $(a,b)_{\in}\overleftrightarrow{\mathbb{K}}^i_{u,v}$ to the discrete

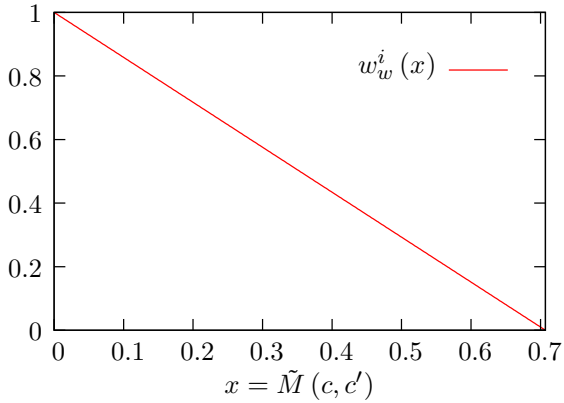

Figure 2.20. Weighting function of the per pixel forward mapping. The colour information is distributed in the target area within a distance of $1/\sqrt{2}$ in the neighbourhood of the true warping position.

pixel target position $c = (u, v)$, is measured as

$$\tilde{M} : (U \times V) \times \mathbb{R}^2 \mapsto \mathbb{R} \tag{2.8.14}$$

$$\tilde{M}(c, c') = \|c - c'\|_2. \tag{2.8.15}$$

By distributing the warped colour information, the accumulated weights w_w^i have to be buffered, as well as the accumulated colour, in order to normalise the weighted sum after all source pixels have been warped. The weights are defined by a linear mapping of the distance, limited by the radius of a pixel as

$$w_w^i(u, v) := \begin{cases} 1 - \sqrt{2} \cdot \tilde{M}(c, c'); & \text{if } 0 \leqslant \tilde{M}(c, c') \leqslant 1/\sqrt{2} \\ 0; & \text{else,} \end{cases} \tag{2.8.16}$$

to the interval $[0, 1]$.

The resulting weighting function is plotted in Figure 2.20. Additionally, the z-buffer test has to be relaxed to allow for neighbouring pixels in the target domain to be accumulated within a given disparity range.

When the disparity of the target view is available, an inverse mapping, or backward mapping, can be applied. In contrast to the forward mapping, the sampling is done in the target image domain, avoiding holes due to quantisation or sampling errors. Given the disparity of the target view C_2, the image content of the corresponding view C_1 can be transferred to view C_2 by

$$I_{C_2}(u, v, \cdot) = I_{C_1}(u + \lfloor D_{C_2}(u, v, 0) \rceil, v + \lfloor D_{C_2}(u, v, 1) \rceil, \cdot) . \quad (2.8.17)$$

Image warping allows the transformation of one image, utilised in this application to form a novel view. However, aside from artefacts introduced by the warping technique, e.g. sampling and quantisation errors, there are artefacts that originate from the change of viewpoint and the geometry of the scene, e.g. occlusions, that can not be remedied without additional information.

2.9 Criteria Related to Multi-View Displays

The last section introduced the idea to use image warping and corresponding depth maps to render novel views. Depth information provided the information about the scene that is essential to establish the relation between the source and target image. However, image warping itself is limited to one input image and therefore lacks the possibility to yield view interpolations of higher quality. DIBR extends the concept of view interpolation to multiple input views, and provides the opportunity to obtain better interpolation results by joining the information of several input views in the result.

DIBR was, and still is, an active field of research that covers all kinds of different applications. In the following, the field of application is restricted to view interpolation for multi-view displays based on lens arrays. The requirements that are derived for this special application partly conform to the classification introduced by Buehler et al. [BBM+01], whereas other requirements become negligible, due to the structured viewpoints and uniform camera projection.

2.9.1 Viewpoint

For multi-view displays all light rays pass through a plane, which is defined by the physical display plane. Furthermore, the light rays are restricted to the active parts of the display plane, i.e. the spatial dimension of a pixel and the angular dimension each pixel emits light into. As a consequence, the position of perspective images that are rendered for a multi-view display are constraint to a plane with the viewpoints located at the optical centres of the lens systems of the multi-view display.

2.9.2 Occlusion

Another aspect that has to be considered, when images are interpolated from a sparse set of images, is occlusion. Occlusion occurs when a foreground object blocks the view of a part of the scene, as pictured in Fig. 2.21 on the left side. If the dashed view is reconstructed from the solid drawn view, the

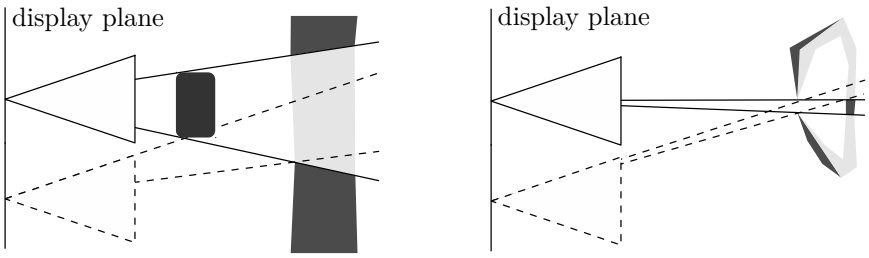

Figure 2.21. Occlusion (left) and self occlusion (right) caused by objects in the scene.

background object will have some parts missing. Occlusions have the most severe impact on image interpolation if a part of the scene is only visible in the view that is reconstructed but not in any of the input views. Therefore, no input view can be used to fill in the missing image information.

A case of self occlusion is pictured in Fig. 2.21 on the right side. In this special case the concave object itself occludes the view, resulting in a partial occlusion that can not be reconstructed by the neighbouring views.

Occlusions can be arbitrary complex and depend on both, the arrangement of the objects in the scene and the geometry of the objects. In the worst

case, it is not possible to omit views without introducing severe artefacts, due to occlusions or self occlusions, in the view interpolation.

2.9.3 Sampling of the Scene

Artefacts, that are similar to the artefacts of occlusion, can occur when the input images for the interpolation algorithm does not sample all objects in the scene. A subsampling of the scene with perspective input images is depicted on the left side of Figure 2.22. The input images are drawn solid

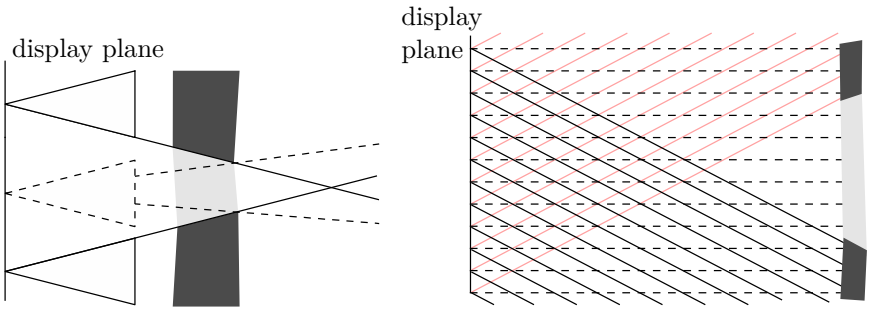

Figure 2.22. Subsampling of the scene with perspective views (left) and orthographic views (right), leading to a artefacts in the interpolated view (dashed lines).

and the interpolated image is drawn drafted. Subsampling from perspective input images depends on the opening angle of the perspective views and the spatial distance to the next input image. With decreasing distance to the display plane, objects are subject to subsampling when interpolation is done from a sparse set of perspective input images.

The right side of Figure 2.22 shows a subsampling of the scene with orthographic input images. For orthographic input images, subsampling of objects occurs at an increasing distance to the display plane. Due to the parallel viewing rays, the point where the scene is not sampled depends on the angular distance of the input images and the spatial resolution.

2.9.4 Minimum Angular Deviation

For most surfaces with Lambertian reflectance, view interpolation yields good results, as long as the surface is visible in at least one input view. For reflective surfaces, an additional criteria has to be taken into account, the angular deviation.

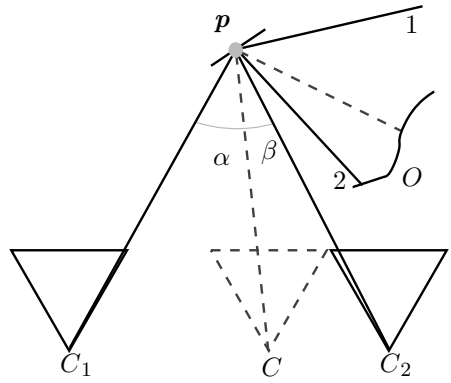

Figure 2.23. Reflection of an object on a glossy surface.

Figure 2.23 shows a scene with a reflective surface point $\boldsymbol{p}$, an object O, and two views C_1 and C_2, that are used to interpolate another view C. The object is reflected in the surface point $\boldsymbol{p}$. To interpolate view C, the angular closest light ray should be used, i.e. the light ray of view C_2, as the angle β is smaller than the angle α to the light ray from view C_1.

The reason is that the colour of a reflective surface point depends on the angle under which it is observed. Without a priori knowledge, it is assumed that the smaller the angle between the interpolated viewing ray and the input viewing ray, the smaller is the interpolation error. In Figure 2.23 view C_2 and C sample both the back side of object O via point $\boldsymbol{p}$, whereas the viewing ray of view C_1 misses the object.

Some scenes do not have object reflections on surfaces but highlights instead. Highlights can be seen as reflections of a light source on a surface, therefore a minimum angular deviation will also yield better interpolation results. During interpolation, the surface properties are unknown, therefore a minimum angular deviation should be favoured in general.

2.9.5 Small Motion Consistency

When the observer of the multi-view display moves by a small amount, the changes in the observed view should reflect the small viewpoint change by a smooth transition. This holds for both, the angular and the spatial resolution. Figure 2.24 shows an observer that moves by a small amount from C to C'. In the centre of the figure are three lens systems of the display,

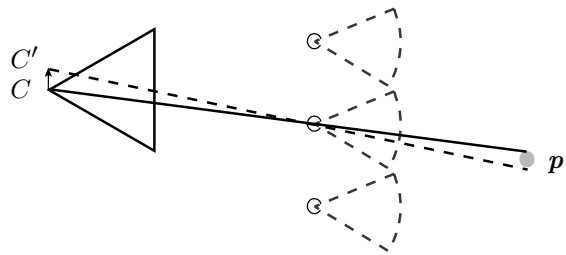

Figure 2.24. Small motion consistency for angular resolution.

with their opening angle sketched by a dashed line. The view dependent colour information can be thought of as a picture at the outer line of the sketched opening angle, sampled by the viewing rays. The scene shows a virtual point $\boldsymbol{p}$, placed behind the display and observed by the camera at the positions C and C'. If the observer moves by a small amount, the viewing ray that passes through the lens system of the display samples a slightly different location in the image that encodes the view dependent colour information.

A display with a high angular resolution will respond to the viewpoint change with a smooth transition, as the viewing ray samples the virtual point $\boldsymbol{p}$ at a different location.

Figure 2.25 shows again an observer moving a small amount, but this time, a display with a high spatial resolution. The motion of the observer from C to C' causes a parallel shift of the viewing ray that now samples a neighbouring lens system of the display. The parallel shifted viewing rays still sample an area close by the virtual point $\boldsymbol{p}$. Again the observer should see a smooth transition while observing the virtual point.

It follows that the area around a point in the scene should not only be interpolated with smooth transitions in the view dependent colour information, but also in the spatial neighbourhood to avoid sampling artefacts

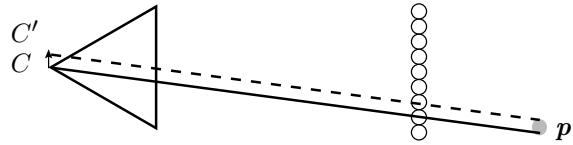

Figure 2.25. Small motion consistency for spatial resolution.

during observation.

Effects that violate this goal are sampling artefacts like aliasing. It is not sufficient to apply this requirement to the interpolation only. Furthermore it has to be taken into account during model creation because a high resolution of textures or a detailed geometry can violate the goal of small motion consistency, even when no interpolation is applied. This is the case when the display is not capable of properly displaying the content due to its spatial or angular resolution.

In general all lens based multi-view displays reduce spatial resolution (see Lanman et al. [LWH+12]) and are therefore prone to aliasing for content near the display plane. It can be remedied by avoiding high frequency in the modelled scene or by a depth dependent low-pass filtering of the elemental images, for content near the display plane.

2.10 Conclusion

The increasing number of views supported by emerging autostereoscopic multi-view displays represent a challenge to the rendering, storage, and transmission of the immense amount of data. In Section 2.7.2 the consequences of the increased data rates were discussed. It was concluded that the most effective way to respond to the increased data rate is to reduce the amount of data at the beginning during rendering, avoiding storage and transmission of the complete data set.

For 3D video displays, the consequence is that the complete data set has to be interpolated on the device in real-time, out of the reduced input data. Due to the large amount of data, the interpolation should be processed in parallel on dedicated hardware. It was deduced that the used interpolation algorithm should access the input in a structured fashion, avoid branching, and allocate few resources in order to map well to the stream processing

model.

The requirements for a DIBR algorithm targeted at view interpolation for multi-view displays was discussed in Section 2.9. The introduced DIBR algorithms will be designed to meet these requirements. Further, consequences for the content creation of multi-view displays were derived that originate from the technical specification of the display, e.g. the spatial and angular resolution.

Although, there has been a lot of progress in the field of IBR and DIBR, none of the previous work fulfils all of the special requirements of this application. In the next chapter the data sets are introduced. For evaluation a simulator for multi-view displays is designed, allowing to modify the technical specifications of the display and render views of a virtual observer. The simulator is further used to render the motion path of an observer in front of the display. This allows to compare virtual views of the interpolated content against views of the ground truth content, which are more meaningful for application then comparing the elemental images of the single lens systems.

Chapter 3

The Data Sets

The creation of data sets for multi-view displays with a high angular resolution is very costly. This is mainly due to the high number of views being rendered, adding to the costs of model creation. In the following, two data sets are introduced that are used later on for evaluation. For both data sets all colour and all depth images were rendered, allowing for an evaluation against ground truth from interpolation on a sparse input data set.

3.1 Ray Traced Data Sets

Both data sets were rendered with a ray tracer, with a custom camera plug-in that implements the projection model of the lens system, an equidistant fisheye projection (see Section 2.4.6).

The corresponding depth maps could be directly rendered with the modelling tool and were exported to the OpenEXR format, which supports 32-Bit and 16-Bit floating point formats (see Ström et al. [SWR+08]). Orthographic images and depth maps were generated from the perspective ones, according to Section 2.4.7 et seq.

3.1.1 Coffee Capsules Scene

The first data set shows a couple of coffee capsules, floating in mid-air in front of a uni-coloured background. In the remainder this data set will be referred to as the *Coffee Capsules* data set. An orthographic view of the display is shown in Figure 3.1. The background colour has a gradient, dependent on the viewing direction, therefore the gradient is not visible in the orthographic views.

The scene is challenging because of the many small objects that occlude each other and the specular reflections on the metallic surfaces of the coffee

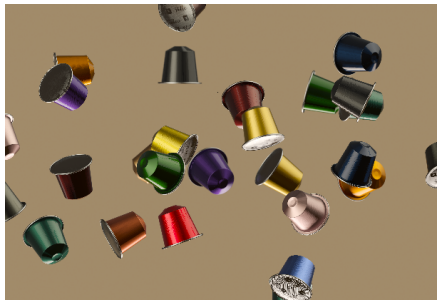

Figure 3.1. Orthographic view of the *Coffee Capsules* data set. The light rays are orthogonal to the display plane. The resolution is 604 x 414 pixels.

capsules. Another challenge for an interpolation are the small letters, printed on the lid of the coffee capsules. The display is placed approximately in the centre of the coffee capsules, such that the coffee capsules are placed from about one meter in front to about one meter behind the display plane. The data set was rendered for a display with a resolution of 604 x 414 pixels and 512 x 512 views, distributed over a field-of-view of 40 degree.

3.1.2 Tutankhamun Scene

The second data set is the mask of Tutankhamun, placed in a large hall. In the remainder this data set will be referred to as the *Tutankhamun* data set. An overview of the scene is given in Figure 3.2, with a view, showing the position of the display in relation to the scene on the left side, representing the display by a red rectangular, and a rendered overview of the scene on the right side. At the end of the hall is a window that is rendered with a bloom post effect from the modelling tool. The distance of the window to the mask is about 25 meters. The scene is challenging due to the reflecting surfaces, especially the ground, and the high level of geometric detail of the pillars and the mask. Adding to the high geometric detail is a high resolution texture on the mask and a golden ornamentation with specular reflections.

The data set was rendered for a display with a resolution of 640 x 360 pixels and 512 x 512 views, resulting in a very high angular resolution. The views are projected equiangular by an equidistant fisheye projection with

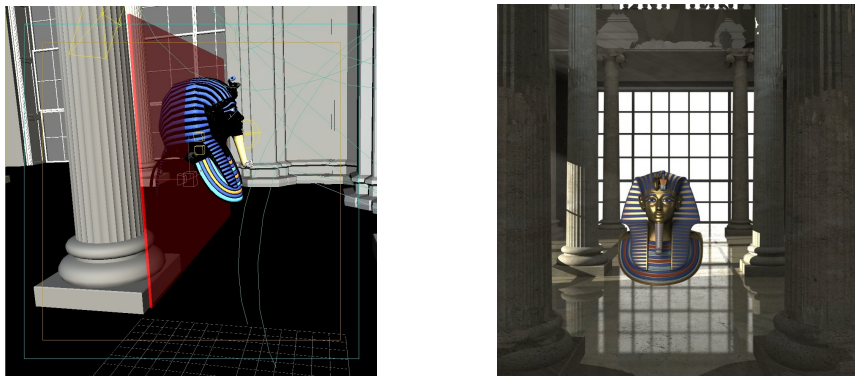

Figure 3.2. Overview of the *Tutankhamun* data set from the modelling tool (3ds Max®, left) and rendered overview (right). The virtual display is placed in the centre of the mask, visible as a red rectangle in the overview on the left side.

an opening angle of 20 degree, therefore only about $\Pi/4$ of the views are used. An orthographic view of the display is shown in Figure 3.3.

A part of the data set had to be re-rendered at a later point. For that rendering the extension that provided the bloom effect at the background window had been unavailable. The result is that for a part in the lower left of the display, the grid of the window in the background is darker and appears thicker than in other areas of the display.

Figure 3.3. Orthographic view of the *Tutankhamun* data set. The light rays are orthogonal to the display plane. The resolution is 640 x 360 pixels.

3.2 Simulation of a Multi-View Display

With the full data set available, it is possible to interpolate the full data set from a sparse set of input images and evaluate the result against ground truth. The direct approach is to directly use the perspective elemental images of the lens systems or the orthographic representation to measure the fidelity of the interpolation. The drawback of this approach is that the evaluation is done on images that are only visible at extreme viewpoints that do not matter for an observer. For the perspective elemental images the viewpoint of the observer would be at the projection centre of the lens system or at an infinity distance to the display, for orthographic views. Another drawback is that a direct evaluation on the images of the data set does not allow for a possibility to evaluate view consistency, i.e. changes in the quality of the interpolation while the observer moves in front of the display.

This drawback can be avoided by simulating the observed views along a defined trail in front of the display. Hence, the evaluation is done on simulated views from the ground truth data set against the simulated views from the interpolated data set. As error metric the AD and the PSNR (see Section 2.5) have been utilised. A moving observer is realised by rendering a sequence of views, where the observer moves along a given path.

The interested reader is referred to Appendix B, where the simulator for the autostereoscopic full parallax display is explained in more detail.

In the following the simulated views will be rendered with a FOV of 20 degree and without distortion. The images of the data sets will be used without any pre-filtering. Deviations of this standard configuration will be mentioned in the text.

3.3 Reduced Data Sets

An evaluation based on the full data set suffers from the drawback that the computation of the results takes a lot of time. An even bigger challenge is the size of the interpolated results that have to be stored and evaluated. For this reasons the size of the data set has been reduced for some evaluations. Reducing the size of a data set can be achieved by selecting a region of interest, by reducing the spatial resolution, or by reducing the angular resolution of the data set.

3.3.1 Selecting a Region of Interest

When a region of interest was evaluated, the resolution of one view has been reduced by deleting all rows and columns of the data set that do not belong to the region of interest. The result is a data set for a display with a reduced size that has the same properties as the original data set, with regards to the region of interest, as the size of the display is reduced by the same factor as the resolution of one view. Therefore the display has been reduced in size, without reducing the spatial resolution of the original display.

3.3.2 Reducing the Spatial Resolution

The spatial resolution of the display can be reduced by reducing the resolution of one view without reducing the size of the display. The spatial resolution is reduced by selecting every n^{th} row and column of the data set, still allowing to view the full scene. For interpolation, the pixel pitch of the spatial resolution has to be multiplied by n to account for the unmodified depth maps. Hence, the result is a data set for a display with a reduced spatial resolution that shows the same scene as the original display, but might introduce even more aliasing.

3.3.3 Reducing the Angular Resolution

In order to evaluate a display with a smaller number of views, the angular resolution has to be reduced. This was achieved by directly resizing the elemental images of the data set. To avoid aliasing artefacts, due to the reduced angular resolution, a 2D Lanczos filter (see Duchon [Duc79]) has been applied for the down-sampling process.

The different methods to reduce the size of the data sets can be combined to simulate different display configurations and fit the desired evaluation goal. The individual configuration of the data set will be given at the evaluation.

3.4 Viewpoints for Evaluation

For the evaluation, a series of viewpoints in front of the virtual display were generated to simulate a moving observer. This path is used in the

remainder to generate all results and to compare the different interpolation techniques. This approach does not compare the whole data set but only the light rays used in the simulated views, but is justified by the advantage that the evaluation is done on views that are relevant to a potential observer.

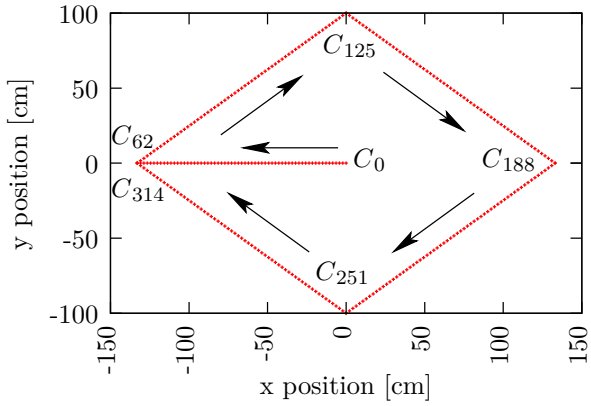

Figure 3.4. Trajectory of the viewpoint positions of the evaluation. The distance to the display is set to 6 meters for all viewpoints.

The path of the observer is shown in Figure 3.4 and consists of 315 positions. The axis of abscissae shows the horizontal deviation from the centre and the axis of ordinate shows the vertical deviation from the centre of the display. The first position C_0 is placed centred before the display. The observer then moves to the left side, afterwards following a rhombus like path in counter clockwise direction. The last position closes the loop of the rhombus at position C_{314}.

The path has been generated from the six corner positions of the plotted path. The viewpoints between the corners were generated using component wise linear interpolation for the 3D position.

The observer's viewing direction has been interpolated using Spherical Linear Interpolation (SLERP) (see Shoemake [Sho85]). For the case that the observer looks at the centre of the display, that viewing direction has been calculated for every position. The standard case for the evaluation is that the observer's view is always directed at the centre of the display.

Chapter 4

Interpolation of Orthographic Images

The following chapter will focus on the viewpoint interpolation using a subset of the full light field in an orthographic representation as input to interpolate the full data set. For every input image, a corresponding depth map will be available, containing the depth information for every pixel of the input image.

The goal of this chapter is to derive the consequences of an orthographic representation of light fields on viewpoint interpolation. Therefore, a basic interpolation algorithm has been implemented, which is described in the following, followed by an evaluation of the interpolation results. The chapter ends with a conclusion about the properties of the orthographic representation and its suitability for viewpoint interpolation for multi-view displays.

4.1 Viewpoint Selection

In Section 3.3, different ways to reduce the size of a data set have been introduced. These ideas can be utilised to reduce the amount of perspective elemental input images for the interpolation. However, the drawback of the proposed approaches is that the subsampling is done on planes parellel to the image plane, insuring equal distances between pixels parallel to the axes of the image coordinate system. For orthographic view directions these methods do not allow to select an arbitrary number of evenly distributed input images out of the domain of all view directions, resulting in different angles between input views that are parellel to the axes of the image coordinate system and the remaining ones. This problem is solved by an optimisation that evenly distributes the number of input samples in a unit

circle, which represents the active image area of the equidistant fisheye projection.

As solver, an algorithm introduced by Levenberg [Lev44] and Marquardt [Mar63] has been chosen, which computes the minimum of a function in a gradient descent manner. The problem description and the approach to solving the problem can be found in Appendix C.

4.2 Overview

After a set of input views has been selected, the algorithm for view interpolation is applied. Figure 4.1 gives an overview of the algorithm that has been used for orthographic view interpolation. The evaluation of the simulated

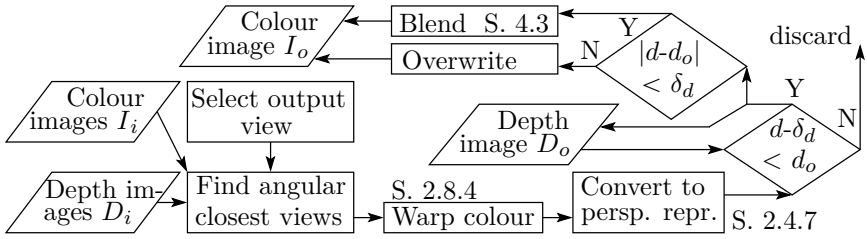

Figure 4.1. Overview of the interpolation algorithm for orthographic images. Disparities are computed with reference to the distance of two neighbouring input images.

views is done after all images of the data set have been interpolated. First, the output view is selected. For the selected output view, the four angular closest input views are determined. All four input views are used in the interpolation, if the target view lies in the centre of the four input views. Depending on the following cases, the interpolation uses less input views. When the output viewpoint lies inside a triangular region of three input views, only the three closest views are taken for the interpolation. In case the output viewpoint lies on a horizontal or vertical line with two input viewpoints, only the two closest viewpoints on that line are used for interpolation, and if the closest viewpoint is the same as the input viewpoint, all other views are omitted for interpolation.

The selected input views are then transferred into the destination viewpoint by pixel wise forward mapping, as described in Section 2.8.4. In order to provide the elemental images for the display device, the orthographic images are transferred into the perspective representation of the elemental images afterwards, as described in Section 2.4.7. Before the interpolated data is written into the output buffer, the visibility test is applied to ensure that foreground objects are not occluded by background objects. Pixel that pass the depth test are written into the output colour image, whereas the depth is saved in the corresponding depth map. Hence, the output depth buffer is initialised with the maximum distance at the beginning of the interpolation of a view. The output colour buffer is initialised with a background colour or a background image. The colour information of the neighbouring views is combined by a weighted blending, which is described in the following.

4.3 Algorithm

In order to interpolate a target view, the colour values of up to four input views are transferred, as described in Section 2.8.4, according to Equation 2.8.10. The result of the interpolation of the different input views is combined by a weighted blending, within a small depth interval to account for the limited numerical precision of the depth images.

Before the warping is initiated, for each input image I_i, out of the set of input images $i_{\in}\mathbb{I}$, a weight $w^i_{L_1}$ is computed as the inverse of the L_1 norm between the position of the input view I_i and the target view I_o as

$$w^i_{L_1}\left(I_i, I_o\right) := \frac{1}{\parallel p(I_i) - p(I_o) \parallel_1}; \, p(i) \neq p(t). \tag{4.3.1}$$

The function $p()$ returns the image coordinates that belong to the perspective elemental images of the orthographic view direction that are used instead of the angular difference, due to the properies of the equidistant fisheye projection (cf. Section 2.4.6). Hence, the weight $w^i_{L_1}$ becomes larger for orthographic views with a small angular distance between them, as the spherical image coordinates are directly linked to the view directions (see Section 2.4.6). As a result, input views with a small angular distance are favoured, enforcing the minimal angular deviation criterion, as described

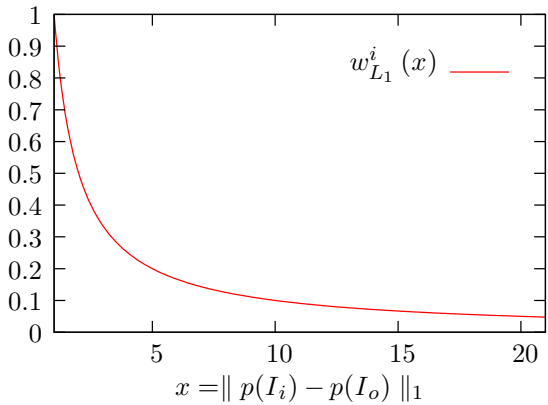

Figure 4.2. Resulting weighting function, based on the distance between the input image I_i and the target image I_o.

in Section 2.9.4. The weighting function is plotted in Figure 4.2. The singularity at $x = 0$ can only occur when an input image is interpolated, which is handled before warping occurs, hence, the minimum distance between an input and a target view is 1.

The colour and depth of each input view I_i is warped into the target view I^i_o, according to

$$I^i_o\left(u + \lfloor D_i\left(u, v, 0\right)\rceil, v + \lfloor D_i\left(u, v, 1\right)\rceil, \cdot\right) = I_i\left(u, v, \cdot\right). \tag{4.3.2}$$

The depth of the input view is converted to disparity according to Equation 2.8.2 in Section 2.8.3. When multiple colour values are warped to the same destination, e.g. at depth discontinuities, the colour value that corresponds to the nearest intersection of the light ray with a scene object is selected by the visibility test.

Due to the relatively low spatial resolution of two millimetres, the forward mapping (see Equation 2.8.16) would lead to large foreground fattening at object borders. Hence, the discrete forward warping is used to limit this effect that becomes even more noticeable when the spatial resolution is further reduced, as can be seen later on in the evaluation.

The set of blended output views $\mathbb{J}$ is defined for every output pixel (u, v)

by comparing the values of the depth test D_o^i in scene coordinates, as

$$\mathbb{J}_{u,v} = \left\{ i \in \mathbb{I} \middle| \left| D_o^i(u,v) - \min_{j \in \mathbb{I}} \left(D_o^j(u,v) \right) \right| < \delta_d \right\}. \tag{4.3.3}$$

The minimum depth value is selected by the visibility test. The depth region δ_d around the minimum depth, which depends on the scale of the scene, is used to account for a limited numerical precision of the depth values, and allows for the pixel wise blending of the warped input images.

The final interpolated view I_o is obtained by weighted summation of the warped input views I_o^i within the depth region δ_d as

$$I_o(u,v,c) = \frac{\sum_{i \in \mathbb{J}_{u,v}} w_{L_1}^i \cdot I_o^i(u,v,c)}{\sum_{i \in \mathbb{J}_{u,v}} w_{L_1}^i}, \tag{4.3.4}$$

normalised by the sum of the blending weights. Due to the discrete forward mapping, the pixels are not accumulated in the neighbourhood around the true 2D warping position and the weight w_w^i is not applied.

4.4 Evaluation

The basic interpolation algorithm of orthographic input images has been evaluated on the *Coffee Capsules* data set, which was introduced in Section 3.1.1, and the *Tutankhamun* data set, which was introduced in Section 3.1.2. For each data set, the full set of elemental images were interpolated from progressively fewer input images. The input view directions were selected out of the set of the 202,963 possible orthographic view directions, as described in Appendix C. A region of interest was selected for the *Coffee Capsules* scene, reducing the resolution to 303×207 elemental images, as described in Section 3.3.1, resulting in a simulated partial display. For the *Tutankhamun* scene, the spatial resolution of the display was reduced to 320×180 elemental images by selecting every 2^{nd} row and column, as described in Section 3.3.2.

The interpolated orthographic views were afterwards transferred into the perspective representation of the elemental images (see Section 2.4.7). Finally, the interpolated light field was used to render the selected viewpoints

(see Section 3.4) of the simulated display (see Section B.2), and the result was compared to the simulated views rendered from the ground truth images.

4.4.1 Scene *Coffee Capsules*

The first evaluation has been done on the *Coffee Capsules* data set. The background of the scene has a colour gradient, which depends on the viewing direction. Hence, a ground truth background image was inserted when the elemental images were assembled, in order to avoid an interpolation error in background areas. The full light field was interpolated 16 times and the number of input viewing rays were gradually reduced from 1.339 percent to 0.186 percent of the full data set. Figure 4.3 shows the PSNR (left) and the AD (right) of the summarised evaluation, plotted against the number of input rays and the simulated position. In this particular scene, it can

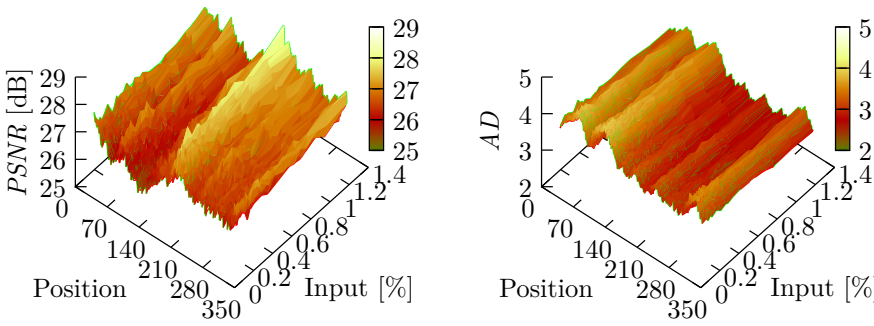

Figure 4.3. PSNR (left) and AD (right) of scene *Coffee Capsules*, plotted against the percentage of input viewing rays and the simulated viewpoints.

be seen that the position of the simulated observer, relative to the display, affects the fidelity of the content more than the number of input rays, in the evaluated interval.

Figure 4.4 shows the PSNR on the left and the AD on the right, with 0.28 percent of all rays as input. The minimum PSNR is at viewpoint Position 110 with 25.38 dB (see top row in Figure 4.4) and the maximum is at Position 176 with 27.65 dB (see middle row in Figure 4.4). The first two rows show the simulated views using ground truth images (left), the interpolated images (centre), and the negated difference between the

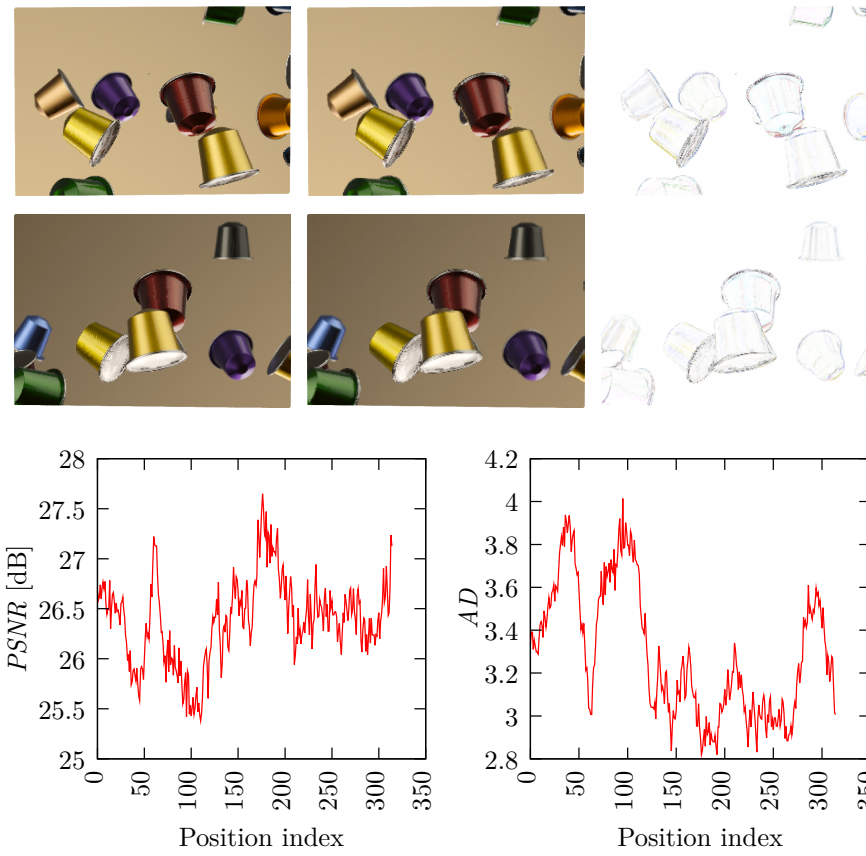

Figure 4.4. Upper row: Simulated view of position C_{110} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{176} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: PSNR (left) and AD (right) of scene *Coffee Capsules* with 0.28 percent of all rays as input. The axis of abscissae shows the viewpoint (cf. Figure 3.4).

simulated ground truth and simulated interpolated views. The errors in the interpolated views are mainly at the object borders and on the printed lids of the capsules. These fine details are lost during interpolation, due

to the relatively coarse resolution of 303×207 of the orthographic views, resulting in obliterated letters on the capsule lids and a fattened capsule border. Prominent artefacts that disturb the overall perception of the scene are not visible.

Figure 4.5 shows the averaged PSNR of all simulated viewpoints, plotted against the percentage of input rays. It can be seen that adding more input images yield only a small increase in the PSNR, in the investigated interval. In the plotted interval, the number of input rays were increased

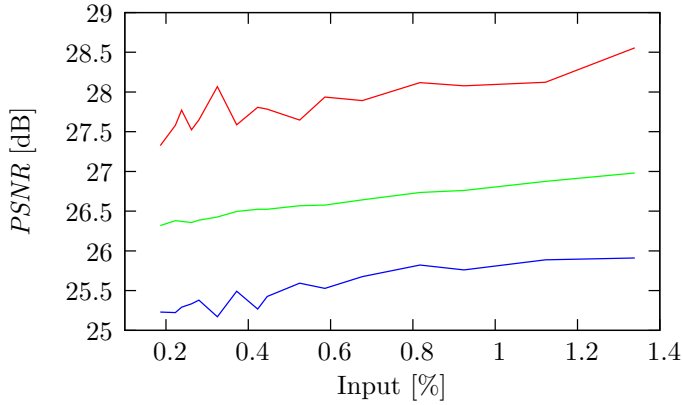

Figure 4.5. Line graph of maximum (top), mean (middle), and minimum (lower) PSNR of scene *Coffee Capsules*, averaged over all simulated viewpoints and plotted against the percentage of input viewing rays.

from 0.186 % to 1.339 %, which equals a factor of 7.20, whereas the mean PSNR of all evaluated viewpoints only increased from 26.32 dB to 26.98 dB, which equals a factor of 1.03.

For the relatively simple warping algorithm and the low resolution of the input images, the interpolation results are promising. The low gradient suggests that the scene is sampled well by the input images, which is supported by the absence of prominent interpolation artefacts. The almost constant image fidelity suggests that the number of input images could be reduced even further, while maintaining an acceptable interpolation fidelity.

4.4.2 Scene *Tutankhamun*

The second evaluation has been done on the *Tutankhamun* data set, as introduced in Section 3.1.2. The background of the scene has a white colour, which was chosen as background colour for the interpolation. The light field was also interpolated 16 times, with the number of input images ranging from 0.194 to 1.348 percent.

Figure 4.6 shows the PSNR on the left side and the AD on the right side, plotted against the number of input views and the simulated viewpoint. An unexpected low PSNR is observed with the maximum number of input images that increases with a decreasing number of input images at first, before proceeding to the expected behaviour of a decreasing PSNR with a decreasing number on input images. That aside, it can be observed that the fidelity of the interpolation is affected stronger by the simulated viewpoint than by the number of input views, which is in accordance to the evaluation of the last section.

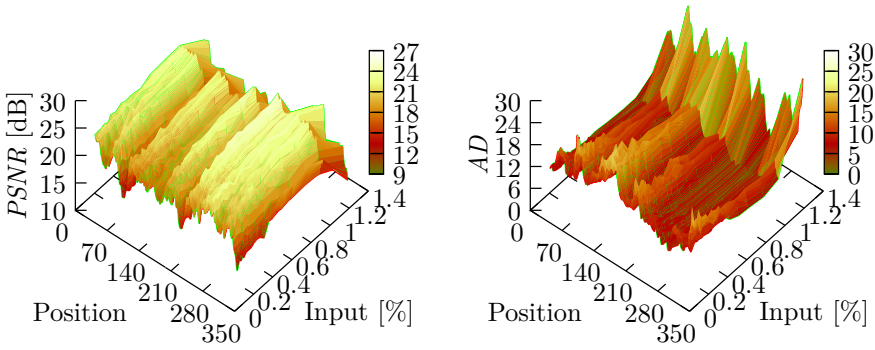

Figure 4.6. *PSNR* (left) and *AD* (right) of scene *Tutankhamun*, plotted against the percentage of input viewing rays and the simulated viewpoints.

Figure 4.7 shows viewpoint C_{62}, which has the minimum PSNR of all interpolated views with the maximum input of 1.35 percent, comparing ground truth (left) against the interpolation with 1.35 percent input images (centre), and against the interpolation with 1.09 percent input images (right). It can be seen that the most prominent interpolation artefacts on the central image are at the background, on the pillar in shape of the silhouette of the mask and at the outer left and right regions of the simulated display, where

Figure 4.7. Simulated view of position C_{62} (minimum *PSNR*, left ground truth, centre interpolated with 1.35 % input images, right interpolated with 1.09 % input images).

the white background colour is visible. These artefacts are greatly reduced or not apparent in the interpolation that used less input views. The reason of this unexpected behaviour is the limitation to the maximum of the four closest input views. With an increasing number of input views, the angle between neighbouring input view directions becomes increasingly smaller, hence increasing the region that is occluded by a foreground object. When an object of the scene lies in that occluded region it can not be interpolated by the closest input views.

This connection is sketched in Figure 4.8. In front of the display lies an object that intersects the green dyed diplay plane. Another Object O is located in the background and is visible by the solid black viewing ray, which should be interpolated from the closest neighbouring views with angle α, drawn as dashed pink lines. One of those views is not suited to sample Object O since all parallel rays, drawn by the stippled magenta line and marked by the orthogonal arrow, which sample the object, lie outside the active display area. Hence, the object is not sampled in that orthographic view. In the other nearest neighbouring view the background object is occluded by the foreground object. Again, the parallel viewing rays are drawn in magenta and marked by an orthogonal arrow. In that view the object is only partly sampled by the solid magenta viewing ray, hence the interpolated view will have a hole in the background object, as can be seen in Figure 4.7. The easiest way to remedy the problem is to use more than the four closest input views, but at the costs of computational power and an increasing memory bandwidth. The occlusion of the background object does not occur when the number of input views is reduced and thereby the angle between neighbouring input views is increased, as seen in the solid cyan viewing ray that has the angle β to the interpolated view.

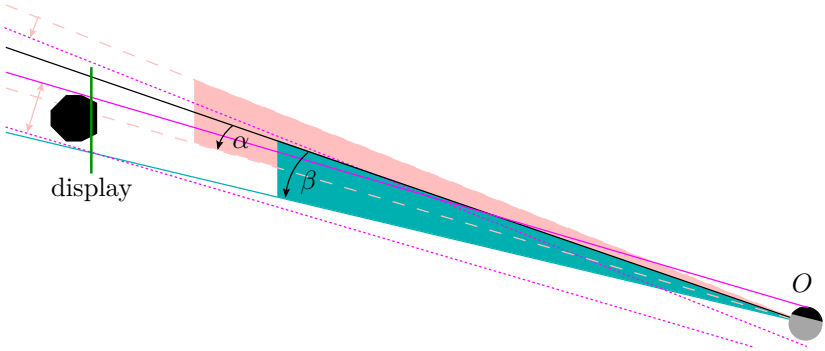

Figure 4.8. Occlusion in orthographic images with many input views and a small angle between neighbouring input views.

Figure 4.9 shows viewpoint Position 310 with the minimum PSNR of 13.40 dB, Position 259 with the maximum PSNR of 23.29 dB, and the plots of the PSNR and AD for all evaluated viewpoints, from top to bottom, of the interpolation with 0.28 percent of all rays as input. The top two rows show the ground truth, the interpolated, and the negated difference images, from left to right. Similar to the evaluation of the *Coffee Capsules* scene, interpolation artefacts are visible at the object borders, in particular noticeable at the border of the mask at the central negated difference image. In contrast to the *Coffee Capsules* scene, whose PSNR lay inside an interval of about 2 dB for all evaluated views, for this scene the PSNR of the evaluated views are in the interval of 10 dB. Comparing the negated differences of the views with the min. and max. PSNR, one notice that the errors on the mask appear to be of similar scale, but the view from Position 310 shows large errors on the background. In the view from Position 259 only a small area of the image shows a background structure and an additional part of the inner surface of the mask, that is occluded in most other viewpoints.

Figure 4.10 shows the averaged PSNR of the simulated viewpoints, in dependence of the percentage of input views. In the evaluated interval, the input views were increased from 0.194 % to 1.348 %, which equals a factor of 6.95, whereas the mean PSNR increased from 17.80 dB to a maximum of 21.47 dB, which equals a factor of 1.21. In contrast to the *Coffee Capsules* scene, the relatively simple warping algorithm causes prominent

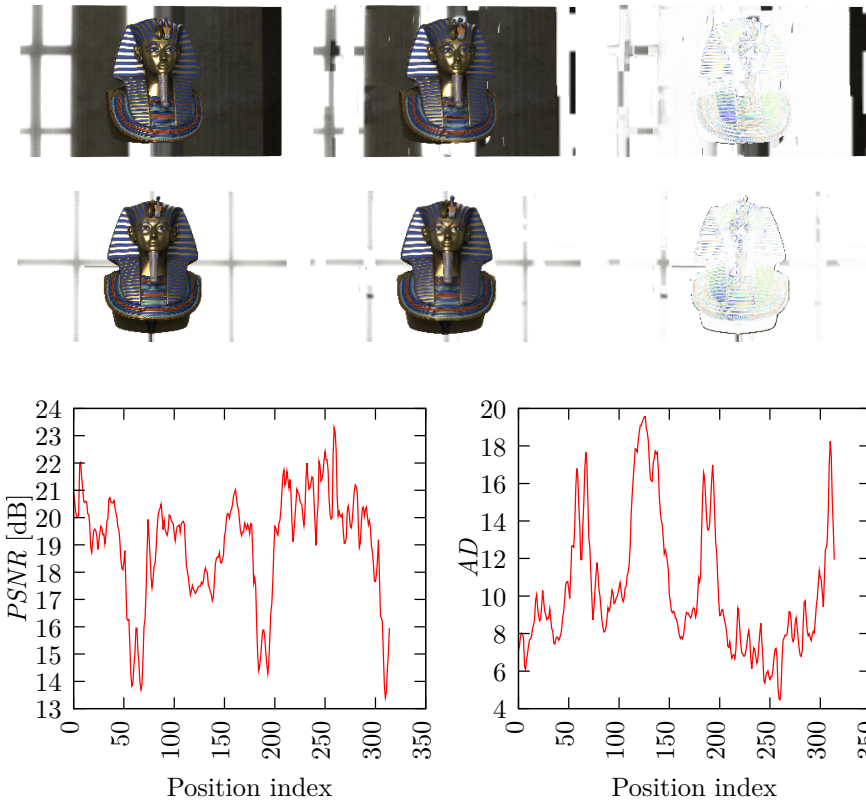

Figure 4.9. Upper row: Simulated view of position C_{310} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{259} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: PSNR (left) and AD (right) of scene *Tutankhamun* with 0.28 percent of all rays as input. The axis of abscissae shows the viewers position (cf. Figure 3.4).

interpolation artefacts in the background area of the *Tutankhamun* scene. Figure 4.10 shows that adding more input views does not solve the issue, but reveals a correlation between the scene and the input views that are required for proper interpolation.

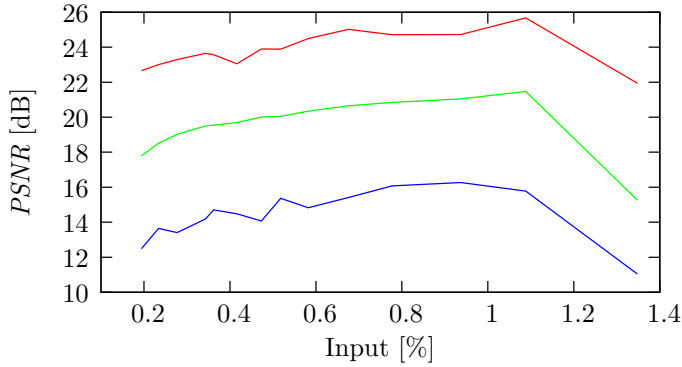

Figure 4.10. Line graph of maximum (top), mean (middle), and minimum (lower) PSNR of scene *Tutankhamun*, averaged over all simulated viewpoints and plotted against the percentage of input viewing rays.

4.5 Conclusion

The evaluation of both scenes showed that the low spatial resolution of the display limits the interpolated image fidelity, which could be expected. That aside, the interpolation of the *Coffee Capsules* data set showed promising results for a relatively simple interpolation algorithm. The initially positive interpolation results of the first data set were put into perspective with the evaluation of the second data set, that showed severe artefacts on background areas. Adding more input views did not solve the issue, but showed that it is not sufficient to interpolate images from the nearest neighbouring views. Depending on the scene, different input views are required for an interpolation without prominent artefacts.

Summarising, this basic DIBR algorithm of orthographic input images worked well on the *Coffee Capsules* scene that consists of many floating foreground objects near the display plane, and on the foreground object of the *Tutankhamun* scene, without introducing prominent artefacts. On the background area of the *Tutankhamun* scene, the flaws of this basic interpolation algorithm could be observed, resulting in prominent artefacts that severely affect the perception of the displayed content. Therefore, this DIBR algorithm is not suited for the interpolation of arbitrary scenes.

Chapter 5

Interpolation of Perspective Images

In this chapter, a view interpolation algorithm is introduced, which is similar to the one in the previous chapter, but uses computer generated light field samples in a perspective representation, and the corresponding depth maps.

The last chapter focused on view interpolation from a subsampled set of view directions, hence, the input had a reduced angular resolution. The input images in this chapter have the full angular resolution but are a subset of the elemental images of the display, therefore, the input of the interpolation has a reduced spatial resolution.

After the properties of the subsampled data set are introduced, an overview of the interpolation algorithm is given, followed by an evaluation of the effects that originate from the incremental reduction of the input images. The chapter concludes with a discussion of the properties of perspective images and the effects on view interpolation.

5.1 Subsampling of the Data Set

For the perspective image representation, the position of the optical centre is a constant (see Section 2.4.7). In the evaluation the spatial resolution is incrementally reduced, as described in Section 3.3.2, by reducing the number of lines and columns, picked as input images for interpolation.

5.2 Overview

Figure 5.1 shows an overview of the algorithm, which is used for the interpolation of perspective images. The evaluation of the simulated views is

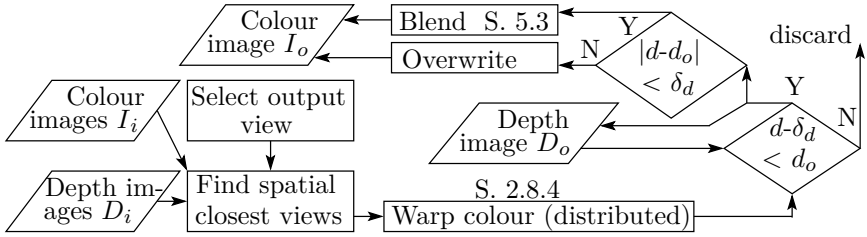

Figure 5.1. Overview of the interpolation algorithm of perspective images. Disparities are computed with reference to the distance of two neighbouring input images.

done after all images of the data set have been interpolated. After selecting the position of the interpolated view, up to four spatial neighbouring input views are selected. Depending of the relation to the neighbouring input views, the nearest four neighbours are selected as input in a rectangular region, the nearest three in a triangular region, the nearest two on a line, and the interpolation is aborted if the target view is available as input, exactly as in Section 4.2.

The selected input views are then transferred into the destination viewpoint, pixel by pixel, and written into the output colour image, whereas the disparity is saved in the corresponding disparity image. The processed data sets have a high angular resolution, compared to the spatial resolution of the data set. Therefore, the colour values are propagated into the neighbouring pixels of the output view, according to the technique described in Section 2.8.4. This will reduce aliasing and one-pixel gaps that may occur due to quantisation during the warping process.

Before the interpolated data is written into the output buffer, it is checked if the current data lies in front of the data that currently is stored in the output buffer, to ensure that foreground objects are not occluded by background objects. Therefore, before any interpolation is done, the depth buffer is initialised with the maximum distance. The output colour buffer is either initialised with a background colour or with a background image. In the following, the blending of the colour information is introduced.

5.3 Algorithm

Due to the high angular resolution of the perspective images, the distributed forward warping, see Equations 2.8.11 to 2.8.16 of Section 2.8.4, is used. The colour is distributed in the spatial 3 x 3 neighbourhood via the blending weight w_w^i, which is defined in Equation 2.8.16.

Again, for each input image I_i out of the set of input images $i \in \mathbb{I}$, the weight $w_{L_1}^i$ is computed, according to Equation 4.3.1 of Chapter 4. In the case of perspective input images, the function $p()$ returns the index position in the spatial domain of the lens systems of the display, therefore favouring small spatial distances with a large weight.

The perspective input images all have the full angular resolution. Near the borders of the active area of the image, rendering artefacts of the aperture occur due to anti-aliasing, which manifest in a loss of brightness. In addition, the viewing rays near the maximum angle of incidence exhibit the maximum vignetting and distortion, which is mainly relevant for photographs, but can also be achieved as a render effect in computer generated content. Therefore, a weight is introduced in order to avoid light rays from the outer image regions, when light rays from the centre are available. The blending weight w_f is reduced for a distance larger than $a = 0.8$, by

$$w_f(u,v) := \left(a \cdot \max\left(\parallel \mathcal{W}^{-1}(u,v) \parallel_2, a\right)^{-1}\right)^4. \tag{5.3.1}$$

The resulting blending function is plotted in Figure 5.2. The weight decreases towards the border of the image, allowing for a smooth blending. The result is similar to the FOV penalty, as described by [BBM+01]. Due to the fisheye projection model and the normalised image coordinates, the distance of each pixel to the centre of the image is limited to the interval $[0, 1]$.

The resulting weight is plotted in Figure 5.3. Inside the inner 80 percent of the FOV the blending weight is not reduced. Near the maximum FOV, the weight is reduced to a factor of about 0.4.

Finally, the output image I_o is interpolated from the set of warped input images I_o^i, where every target pixel (u, v) is supported by a support area

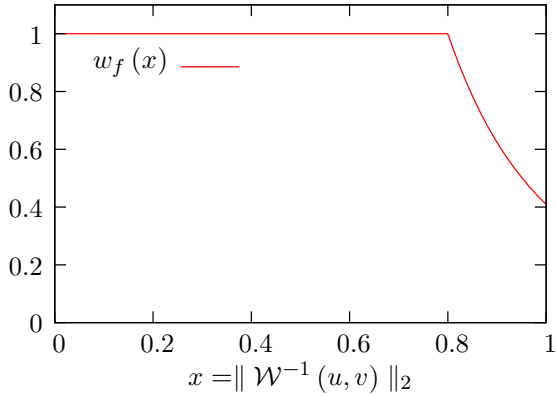

Figure 5.2. Weighting function in dependence of the distance to the image centre.

$\overleftrightarrow{\mathbb{K}}^i_{u,v}$ in the input view i, according to

$$I_o(u,v,c) = \frac{\sum_{i \in \mathbb{J}_{u,v}} \sum_{j \in \overleftrightarrow{\mathbb{K}}^i_{u,v}} w_f \cdot w^i_{L_1} \cdot w^i_w \cdot I^i_o(u,v,c)}{\sum_{i \in \mathbb{J}_{u,v}} \sum_{j \in \overleftrightarrow{\mathbb{K}}^i_{u,v}} w_f \cdot w^i_{L_1} \cdot w^i_w}, \tag{5.3.2}$$

normalised by the sum of all weights.

The product of the weights w_p of Equation 5.3.2 is plotted in Figure 5.3. The axis of the weight $w^i_{L_1}$ is plotted for spatial distances of the interval $[1, 21]$ and the axis of the interval $[-1, 1]$ is used for the weights w_f and w^i_w. Due to the large plateau with a value of 1 and a late decline, starting at 0.8, the FOV blending weight w_f has no visible effect in Figure 5.3, because the weight w^i_w is defined as 0 for values greater than $1/\sqrt{2}$. In Figure 5.3, the weights w_f and w^i_w are condensed on one axis, although, the weights are independent of each other.

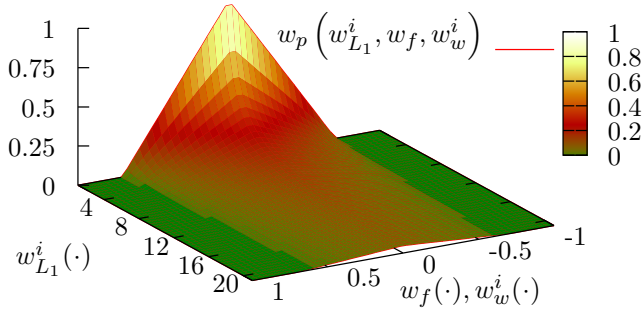

Figure 5.3. Product of the weights of Equation 5.3.2 for blending of the interpolated result of the perspective input images.

5.4 Evaluation

The evaluation of the interpolation of perspective images follows the same structure as the evaluation of the interpolation of orthographic images in Chapter 4. In order to be comparable to the evaluation of orthographic images, the same percentage of input images were used for the interpolation of both scenes for all sampling points of the evaluated interval.

For each data set, the full set of elemental images were interpolated from progressively fewer input images. The input images were selected out of a region of interest of 303×207 elemental images in case of the *Coffee Capsules* scene, and out of the full spatial reduced set of 320×180 elemental images in case of the *Tutankhamun* scene. The input images were selected by further reducing the spatial resolution in an equidistant manner, as described in Section 3.3.2.

Each elemental input image has the full resolution of $202,963$ pixels, corresponding to all orthographic view directions. The interpolated light field was used to render the selected viewpoints (see Section 3.4) of the simulated display, and compared to the simulated views rendered from the ground truth images.

5.4.1 Scene *Coffee Capsules*

In contrast to the interpolation of orthographic input images, no ground truth background image has been inserted for the evaluation of the interpolation of perspective images. With a disparity of zero at infinity, it is expected that the perspective interpolation performs well, even when limited to the maximum of the nearest four neighbours.

Corresponding to Chapter 4, the full light field was interpolated a total of 16 times and the number of input viewing rays were gradually reduced from 1.339 percent to 0.186 percent. The PSNR is depicted on the left of Figure 5.5 and the AD on the right, plotted against the number of input rays and the simulated position. In contrast to the interpolation of orthographic

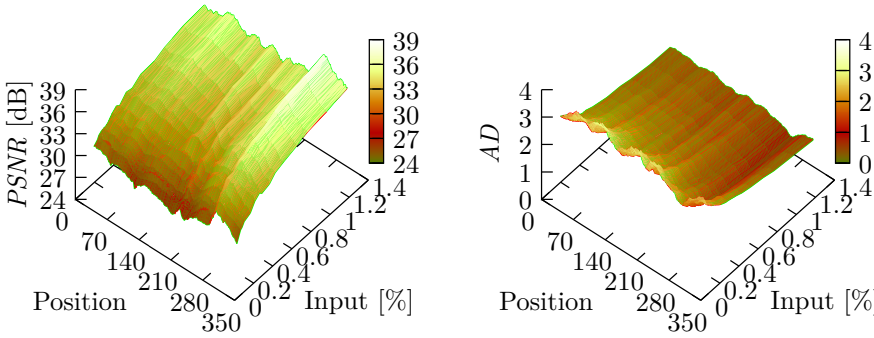

Figure 5.4. PSNR (left) and AD (right) of scene *Coffee Capsules*, plotted against the percentage of input viewing rays and the simulated viewpoints.

images of the last chapter, the position of the observer does not affect the fidelity of the content as much as the number of input views in the evaluated interval. Interpolation artefacts are scene dependent in general. The reason for the different behaviour lies most probably in the different representation of the input data. When the perspective input images are reduced, the distance between overlapping FOVs are increased, directly affecting the fidelity of the interpolation of the foreground content. The foreground content is rarely occluded in any view, hence the fidelity is not that strongly connected to the position of the observer. In contrast, the artefacts of the interpolation of orthographic images are mainly on the background, which is occluded by foreground objects, depending on the position of the observer.

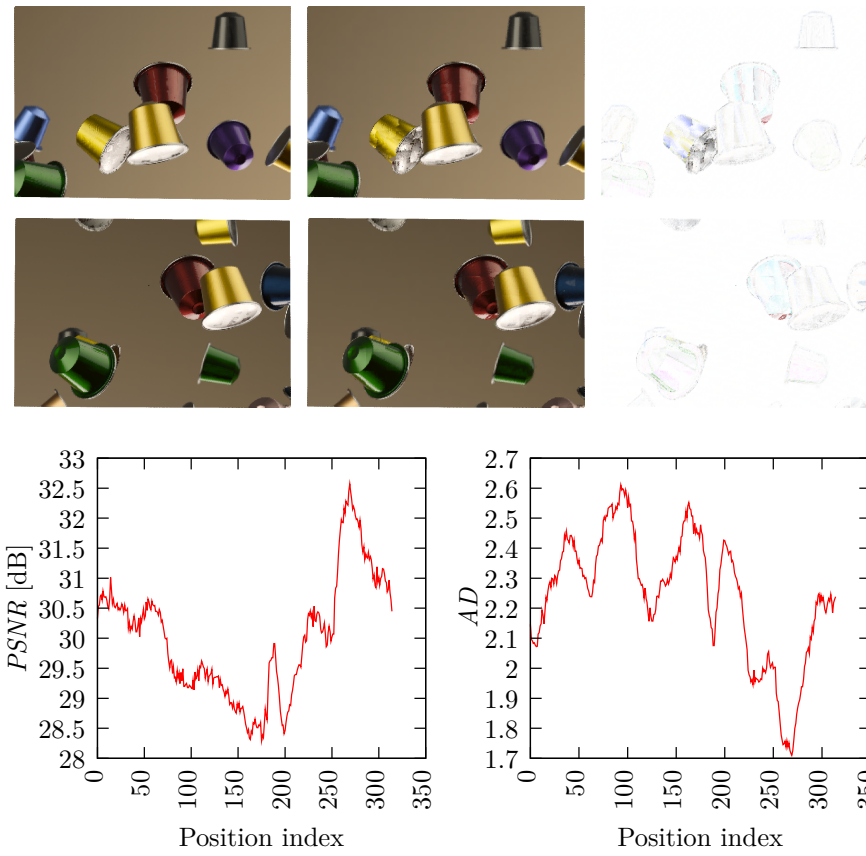

Figure 5.5. Upper row: Simulated view of position C_{175} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{269} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: PSNR (left) and AD (right) of scene *Coffee Capsules* with 0.28 percent of all rays as input. The axis of abscissae shows the viewpoint (cf. Figure 3.4).

Figure 5.5 shows the PSNR and the AD of all evaluated viewpoints at the bottom row, with 0.28 percent of the images used as input for the interpolation. The viewpoint at Position 175 has the minimum PSNR of

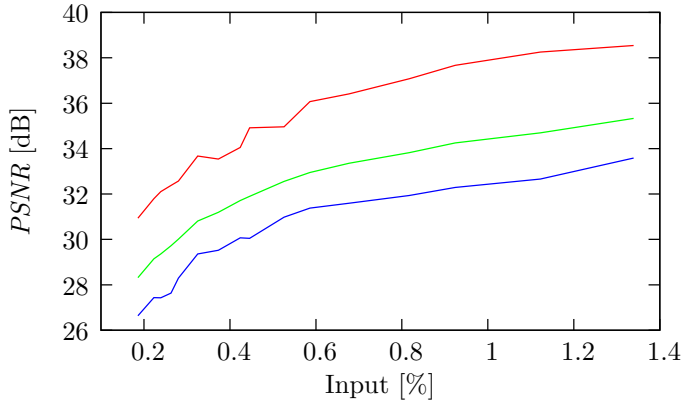

Figure 5.6. Line graph of maximum (top), mean (middle), and minimum (lower) PSNR of scene *Coffee Capsules*, averaged over all simulated viewpoints and plotted against the percentage of input viewing rays.

28.29 dB and is depicted at the top row, showing the ground truth (left), the interpolation (centre), and the negated difference (right). The maximum PSNR of 32.57 dB is at Position 269 and the corresponding images are depicted at the middle row.

The negated difference images show that the errors on the interpolated background are virtually absent, as could be expected. When one compares the negated difference images of C_{175} with C_{269}, it can be seen that the coffee capsule that shows a large error in the interpolated image C_{175} is occluded in the interpolated image C_{269}, by a green foreground capsule. The feature that distinguishes the capsule with the large error from the other capsules, is its close distance to the display plane. Hence, in the perspective elemental images the image information of that capsule has a large disparity, which seems to cause the low fidelity of the interpolation.

Figure 5.6 shows the averaged PSNR of all simulated viewpoints in relation to the percentage of input views. In the plotted interval, the number of input views were increased from 0.186 % to 1.339 %, which equals a factor of 7.20, whereas the mean PSNR of all evaluated viewpoints increased from 28.32 dB to 35.33 dB, which equals a factor of 1.25.

Although the averaged mean PSNR of the perspective interpolation is

much higher than the averaged mean PSNR of the orthographic interpolation, the fidelity of the perspective images shows larger variation with the different viewpoints and is affected stronger by the number of input images. In addition, the appearance of the interpolated perspective images is not of the same uniform fidelity, found in the orthographic images, but shows prominent artefacts on a single capsule in the scene that interferes with the perception of the scene.

In contrast to the orthographic images, finer details could be sustained, resulting in fewer errors at the capsules borders, due to the relatively high resolution of the elemental images.

5.4.2 Scene *Tutankhamun*

The second part of the evaluation shows the *Tutankhamun* scene, interpolated 16 times, with the number of input images ranging from 0.194 to 1.348 percent. Figure 5.7 shows the PSNR and the AD of the different interpolations in relation to the percentage of input images and the viewpoint position. Although the plot shows a large variation of the image fidelity

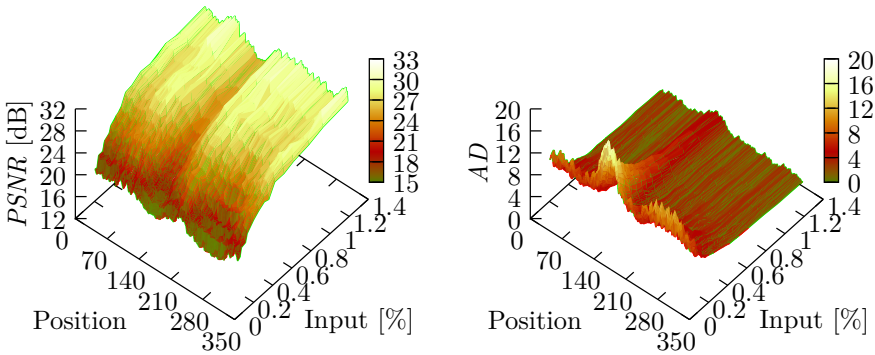

Figure 5.7. *PSNR* (left) and *AD* (right) of scene *Tutankhamun*, plotted against the percentage of input viewing rays and the simulated viewpoints.

for the different viewpoints, this variation seems negligible compared to the loss of fidelity that is associated with the declining number of input images.

Figure 5.8 shows viewpoint C_{257} with the minimum PSNR of all viewpoints of 16.86 dB, viewpoint C_{181} with the maximum PSNR of 22.25 dB,

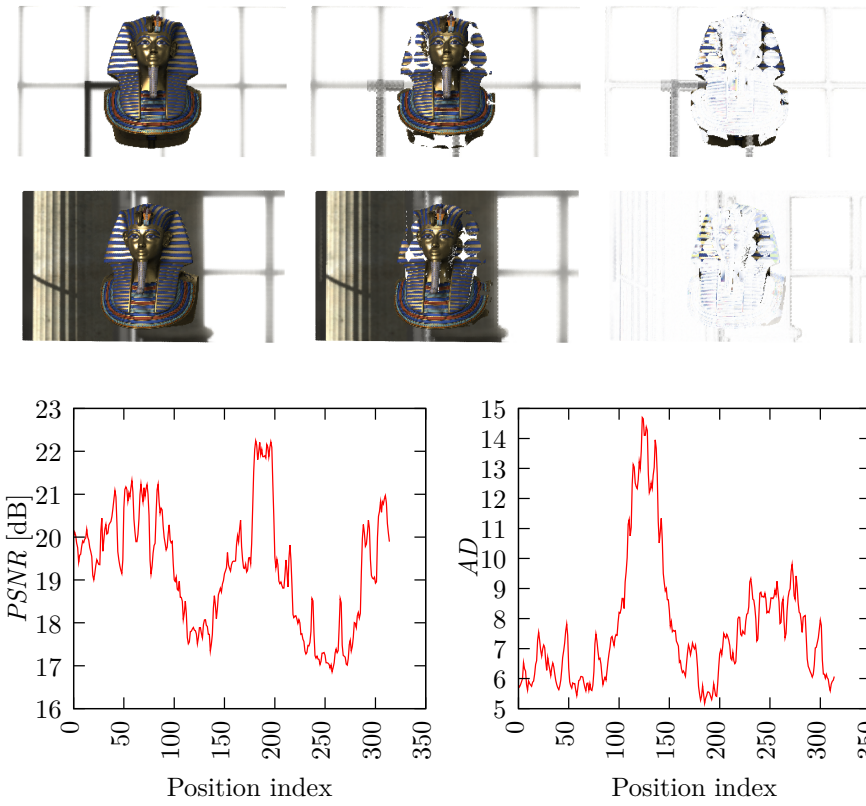

Figure 5.8. Upper row: Simulated view of position C_{257} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{181} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: PSNR (left) and AD (right) of scene *Tutankhamun* with 0.28 percent of all rays as input. The axis of abscissae shows the viewers position (cf. Figure 3.4).

and the plots of the PSNR and AD for all evaluated viewpoints, from top to bottom, of the interpolation with 0.28 percent of all rays as input. The top two rows show the ground truth, the interpolated, and the negated difference images, from left to right.

Figure 5.9. Simulated view of position C_{257} (left ground truth, centre interpolated with 1.35 % input images, right interpolated with 0.19 % input images).

The negated differences show that, aside from the colour dissimilarity on the lattice in viewpoint C_{257} and the edges of the lattice, the background is interpolated well, without prominent artefacts. The same can be noticed at position C_{181}, which shows no prominent artefacts on the pillars and the lattice in the background.

In contrast, the mask in the foreground, which is close to the display plane, has prominent artefacts in the form of holes in the objects outer and inner surface. These holes originate from a non sampled part of the scene, due to non-overlapping FOVs of the neighbouring input images. The problem can be remedied by adding more input views, therefore reducing the baseline of neighbouring input views.

This can seen in Figure 5.9 that shows viewpoint C_{257} for the minimum and maximum number of input views of the evaluated interval. The interpolated view in the centre does not show distinctive artefacts on the foreground mask, and the colour dissimilarity on the background lattice is also reduced, when compared to the interpolation with the minimum number of input images and C_{257} of Figure 5.8.

On the other hand, the artefacts in the interpolated view on the right, with the minimum number of input images, have become more prominent, when compared to C_{257} of Figure 5.8, due to the reduced amount of input images and the larger baseline.

Figure 5.10 shows the mean PSNR of all simulated viewpoints in relation to the percentage of input views. In the evaluated interval, the input views were increased from 0.194 % to 1.348 %, which equals a factor of 6.95, whereas the mean PSNR increased from 17.46 dB to a maximum of 28.56 dB, which equals a factor of 1.64. Hence, the interpolation of perspective images evaluated on the *Tutankhamun* scene has the worst decline in image fidelity

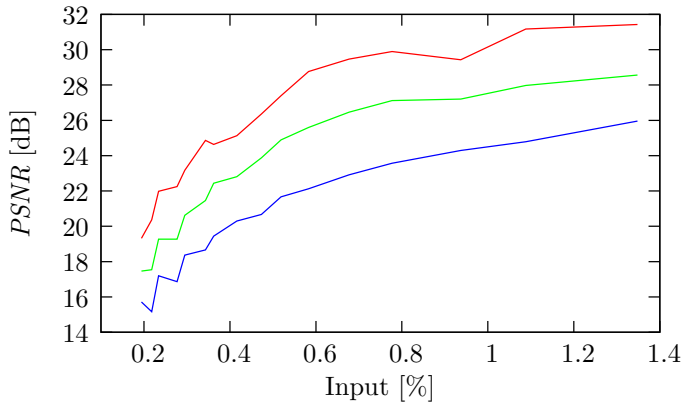

Figure 5.10. Line graph of maximum (top), mean (middle), and minimum (lower) PSNR of scene *Tutankhamun*, averaged over all simulated viewpoints and plotted against the percentage of input viewing rays.

so far, which reflects the prominent artefacts on the foreground mask that severely affect the perception of the scene.

When compared to the *Coffee Capsules* scene, not only the overall PSNR of the interpolation has been lower, but also the decline in image fidelity when fewer input images were used for the interpolation. This is most likely caused by the object arrangement of this scene, which has a large foreground object placed close to the display plane.

5.5 Conclusion

In contrast to the low resolution of the orthographic input images, the perspective input images have a higher resolution of a factor of 3.2 and 3.5, respectively. The orthographic representation is also more prone to aliasing, due to the relatively large distance of the lens systems (cf. Appendix B.4), especially when the data set has been spatially reduced for the evaluation. Together with a more advanced forward mapping, this leads to a significant higher PSNR in both scenes, as can be seen by comparing the mean PSNR of all viewpoints in relation to the number of input views. However, the steeper descent of the interpolated perspective images lead to almost the

same mean PSNR with the minimum number of input images.

In case of the *Coffee Capsules* scene, the interpolated perspective images had in general a higher fidelity, but showed prominent artefacts, depending on the distance of the object to the display plane, which lead to an unpleasant perception of the content and draws the attention of the observer. Although the general fidelity was lower in the interpolated orthographic images, no prominent artefacts occurred, resulting in an undisturbed perception of the scene.

The *Tutankhamun* scene showed prominent artefacts in the interpolated orthographic and perspective images. In case of the orthographic images, the prominent artefacts were located on the background and caused by limiting the interpolation of a viewpoint to the maximum of the four nearest neighbours, as well as too few input images. The perspective images showed prominent artefacts on the foreground mask, which became increasingly severe when the number of input images were reduced. This was caused by non overlapping FOVs of neighbouring input views, that were also responsible for the steep decline in image fidelity and depend on the distance of the content to the display plane. It could be remedied by adding more input views, as was demonstrated in Figure 5.9.

As a consequence, in the next chapter the restriction to interpolate a view from the neighbouring viewpoints is relaxed. Perspective elemental images are solely used as input for the interpolation, because on displays with a low spatial resolution the interpolated images showed a higher fidelity and the conversion between the different image representations is not required. By incorporating a scene representation, the contributed viewing rays of more input images could be utilised for interpolation, allowing to reduce the number of input images in areas where the content is far away from the display plane, hence compensating the increased number of input images in areas where the content is close to the display plane and an overlapping FOV has to be ensured.

Chapter 6

Incorporating a Scene Representation

The content of this chapter has been previously published in Jung and Koch [JK10], and Jung and Koch [JK11b]. In case of differences, this publication shall precede the earlier publications.

In this chapter, a system to interpolate images for the static content of 3D posters is introduced. The goal is to offer a computational advantage, compared to the rendering of all images via ray tracing. The investigation of the last two chapters introduced a couple of challenges and consequences for DIBR using orthographic and perspective data.

Comparing the PSNR of the interpolated orthographic and perspective images, it was observed that the interpolated perspective images yielded a higher PSNR than the interpolated orthographic images, due to the low spatial resolution of the display. Hence, for the interpolation of images for 3D posters, perspective images are utilised, which inherent the benefit that the image representation does not have to be converted, a costly operation, as described in Section 2.4.7.

Another conclusion of the last chapters was that it is not sufficient to deliver the angular closest input views for interpolation, because in those views the occluded regions behind a foreground object are extremely large. Although this applies mainly to orthographic input images, the derived principle to choose from all input data during interpolation holds, because occlusions can become arbitrary complex.

These challenges are met by incorporating a 3D representation of the scene. The idea is to build on a point based scene representation and to interpolate the complete data set out of a few input images. In order to hold a maximum amount of input data in memory, a lossy compression scheme is incorporated, allowing to exclude the redundant data of Lambertian surfaces.

The scenes considered here have very high scene depth and shall display scene depth of many meters, with a lot of foreground objects that will occlude the background in dependence of the viewpoint. For sparse sampling of the plenoptic function, depth information has to be used, or the parallax of the viewpoint to the next plenoptic sample will lead to noticeable reconstruction errors.

6.1 Overview

In the following an outline of the proposed DIBR algorithm is given. The basic idea is to reduce the number of input images by reducing the spatial resolution of the display. Depth maps are available for all colour input images of the reduced data set, which are used to construct a volumetric model of the scene. Therefore, the selection of the input images is the same as in Chapter 5. The difference is that the interpolation process is not limited to the closest four views, but the interpolation of a single elemental image benefits from all input images. The accurate depth images that are required to build a volumetric model of the scene, suitable for image interpolation, limit the proposed algorithm to synthetic scenes or to an application where very accurate depth information is available.

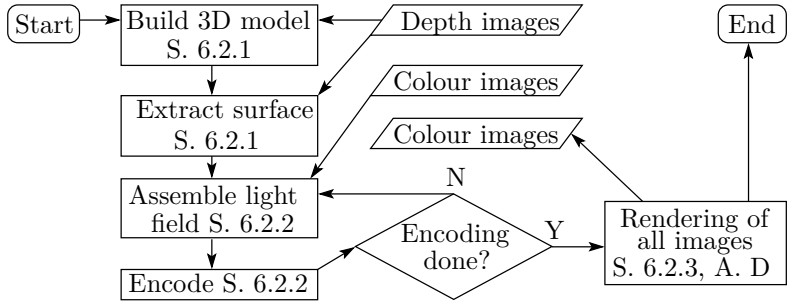

Figure 6.1. Overview of the scene based interpolation algorithm.

Figure 6.1 shows an overview of the proposed interpolation algorithm. The geometry of the scene is represented by a volumetric 3D model that is build from the depth images. Afterwards, the surface is extracted, purging all occluded 3D points from the data structure. The colour images are used

to add colour information to the model in the next step. A lossy compression algorithm is introduced to discard redundant colour information and reduce the memory consumption of the volumetric model, as the light field is assembled in an iterative manner. After the 3D model is build and the colour is assembled, all elemental images are rendered.

In order to handle very large and detailed scenes, it is vital to efficiently handle sparse data and adjust the spatial and the angular granularity. Therefore, the selection of data structures is important and will be discussed in detail.

6.2 Algorithm

Some parts of the algorithm have been parallelized to obtain a significant speed-up compared to serial execution. The parts that have been parallelized on the CPU affect the building of the 3D model, extracting the surface, assembling and encoding of the light field, and rendering. The computationally most expensive parts of the algorithm that are needed for surface extraction, assembling of the light field, and rendering, have been further accelerated by parallelization on the GPU. The implemented parallelization techniques are described in Appendix D in detail.

6.2.1 Building the Geometric Model

The first stage of the algorithm is to build a geometric model of the scene. The model is built from a set of depth images of the scene that can easily be rendered by any modelling tool. This avoids the problem of exporting the geometric model from different modelling tools, and provides a convenient and generic interface. The virtual display plane is a plane in the virtual scene, defined by the location of the multi-view display. The camera centres of all views are located on that plane, and an observer of the display will focus on that plane of the virtual scene, when focusing on the display. For every depth image and for every pixel of each depth image a ray cast is done, in which the depth value defines the length of the ray. The result is a scene description in form of a 3D point cloud.

Figure 6.2 (left) depicts the 3D point representation where $\mathbf{V}$ is the position of the point, relative to the model's coordinate system, according to the five-dimensional plenoptic function P_{5D} (Equation 2.3.2). The sphere,

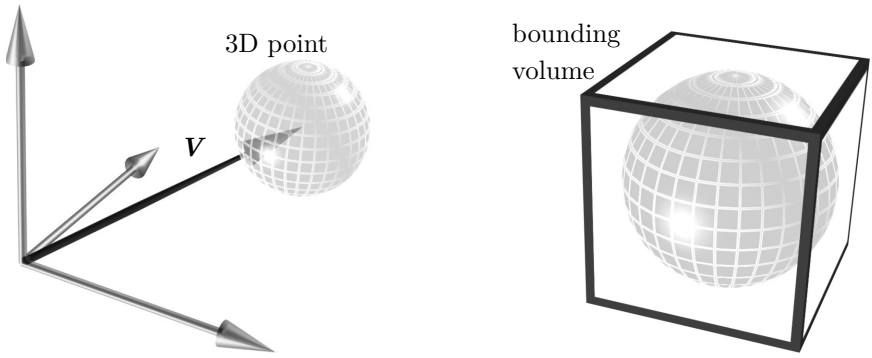

Figure 6.2. Representation (left) and bounding volume (right) of a 3D point.

which is drawn around the 3D point in Figure 6.2 (left), illustrates the storage capacity that is needed to save the different rays that may pass through the 3D point.

The 3D points are stored in an octree. An octree is a hierarchical data structure that partitions 3D space and is able to handle sparse data efficiently. The root node of the octree defines a cube that encloses the complete scene. When a 3D point is inserted into the octree it gets a size assigned that will define at which hierarchical level of the octree the point will be inserted. Figure 6.2 (right) shows a 3D point that is placed at the centre of its bounding volume. Starting at the root node, it is checked recursively, in which sub-volume of the root node the point lies, and the 3D point is inserted in this child of the current node, until the volume of the next hierarchical level is smaller than the volume assigned to the 3D point. If the volume, where the 3D point is inserted, is empty, a new node is added, otherwise, the point is merged with the existing 3D point.

The volume assigned to the 3D point depends on the distance of the point to the virtual display plane. The size of the cube limits the geometric detail that can be described by the geometric representation. Near the virtual display plane the size of the cubes is chosen to be very small, because near the camera centres the viewing rays allow a very high spatial resolution. Far away from the virtual display plane the viewing rays diverge, limiting the spatial resolution and allowing for a much coarser geometric model.

The situation, that 3D points are merged, occurs often because of the

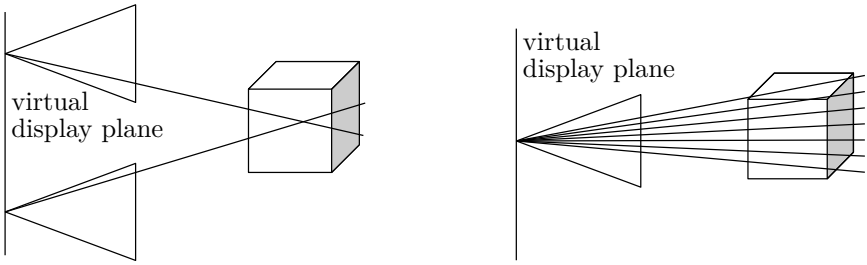

Figure 6.3. Overlapping *FOV* (left) and 3D point near the virtual display plane (right).

overlapping FOVs of the input images (Figure 6.3, left). This will also occur near the virtual display plane, as the different rays are close together near the camera centre (Figure 6.3, right).

Surface Extraction

After the depth images are added to the model, the visible surface of the model is extracted. Due to the limited numerical precision of the depth images, a surface of the scene may be thicker than one layer of volumetric 3D points in the octree representation (Figure 6.4). These additional 3D points lie directly under the surface and are occluded in all views. A 3D scene contains a huge amount of 3D points, typically between 250,000 and 550,000. Therefore, it is beneficial to remove the occluded points, which will save memory and computation time due to the reduced complexity of the model.

In order to remove the occluded points, the model is intersected with all possible viewing rays of the display. When a 3D point is intersected with a viewing ray, it is marked as visible. After all viewing rays have been intersected with the model, all non-marked 3D points are removed from the geometric model of the scene. This ensures that no visible parts of the model are removed without resorting to a priori knowledge of the model's complete geometry.

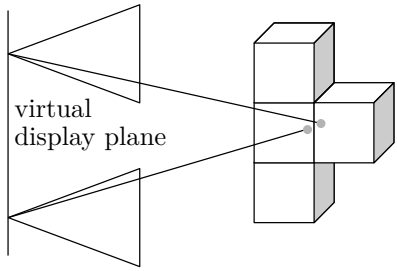

Figure 6.4. Quantisation leading to occluded sub-surface points.

6.2.2 Assembling of the Light Field

After the viewpoints have been selected, the colour information is added to the geometric model. Each 3D point of the geometric model has a data structure attached that holds the view dependent colour information. In the following this data structure is called a Light Field Node (LFN). There are two requirements the data structure has to fulfil that are linked to the properties of the objects in the scene. For objects that exhibit Lambertian reflectance or belong to the background of the scene, where the viewing angle does not change significantly for the different views, it is sufficient to save one colour value per LFN, whereas LFNs near the virtual display plane may need to store more than one colour information per LFN, due to specular reflections. Therefore the data structure of the LFN has to handle sparse data efficiently.

Another requirement is to encode the direction of the saved viewing ray and the possibility to efficiently find the angular closest viewing ray in the data structure for interpolation.

The view dependent colour information is saved in an image of the same properties and orientation as the camera that renders the light field for the display. In this case, it is a spherical image and the direction of the viewing ray is encoded in the spherical coordinates of its position in the image. The LFN can be imagined as a unit sphere with its centre at the position of the 3D point. Therefore, a viewing ray is completely described by the position of the LFN, and the two angles (φ and θ) of the spherical coordinates. See Section 2.4.6 for a detailed description of the projection and the mapping to the data structure.

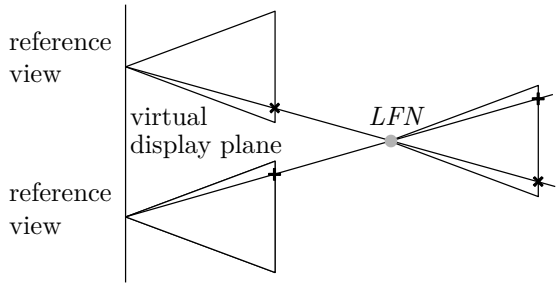

Figure 6.5. Relation of the image index of the reference image and the image index of the *LFN*.

All input images for the display are rendered with a camera of a constant orientation, and its viewpoint restricted to a plane. Under this condition, it is advantageous to orient the LFN the same way as the camera that rendered the input images. This set-up is shown in Figure 6.5. The benefit is that it is not required to project the centre of the camera to the spherical image of the LFN to get the image coordinates and correctly encode the viewing ray's direction, but the image coordinates of the LFN are the same as the image coordinates of the input image. The interpolation of all views will also benefit from this set-up, as the orientation of the interpolated views is the same as the orientation of the input views and all viewpoints are located on the virtual display plane.

Another benefit of this configuration is that the opening angle of the LFN, in which viewing rays can be saved, is the same as the opening angle of the elemental images. Therefore the LFN can only save viewing rays that are in the FOV of the elemental images.

The second requirement is to handle the sparse data of the view dependent colour information, saved by a LFN. The number of elemental images that may contribute to the colour information of a LFN depends on its distance to the virtual display plane. A LFN located in the display plane is visible only in one elemental image but encodes the full view dependent colour information of that elemental image (cf. Figure 6.3, right side). Far away from the display plane, a LFN may be visible in a large number of elemental images but will encode only one view dependent colour information from each elemental image. For good interpolation results for

objects near the display plane the resolution of the view dependent colour information of a LFN is set to the resolution of an elemental image. If every LFN would save a full elemental image, the memory requirement would limit the presented algorithm to about 5,000 LFNs. Typically, the processed scenes have 250,000 to 550,000 LFNs, which is about two orders of magnitude more. Therefore a quad tree (see Finkel and Bentley [FB74]) was chosen, which has several properties that are beneficial for the proposed algorithm. First, a quad tree can save sparse data efficiently, allowing to save as many viewing rays as necessary per LFN, allocating memory only for saved viewing rays. Another advantage is the efficient look-up of angular closest viewing rays, which is important for both, the decision if a viewing ray has new information and should be saved in the data structure, and also for the interpolation of novel views for the display. In the following it is described how the colour information is saved in the LFN.

Lossy Light Field Encoding

The colour information is added to the geometric model by intersecting the model with the viewing rays, and saved in the quadtree of the intersected LFN. When a viewing ray intersects a LFN, it is decided whether the viewing ray should be saved or discarded. Figure 6.6 (a) shows a draft of successive viewing rays that intersect with the LFN, reduced by one dimension for the sake of clarity. The rays are labelled in their initial intersection order with the LFN, and the brightness shall represent the colour of the ray. Figure 6.6 (a) shows which viewing rays are saved after all input images have been processed. When the first viewing ray is processed, the LFN has no colour information saved. Therefore the viewing ray is saved, denoted by a white circle in the figure. When the second ray is intersected with the LFN it has the same colour as the angular closest ray and the viewing ray is discarded. Due to this reason, all following viewing rays are discarded, until the last viewing ray is processed, which has a different colour than the first viewing ray, and is therefore saved by the LFN.

This approach allows to reduce the number of saved rays, which is important for Lambertian surfaces of the scene or when the surface colour does not change within the viewing angle. A drawback is that the interpolation between the angular closest viewing rays during view interpolation will introduce errors for the reconstructed viewing rays between the saved samples, as depicted in Figure 6.6 (b).

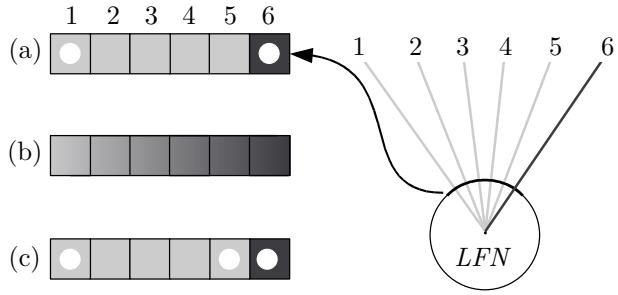

Figure 6.6. Similarity based selection of samples, saved by a *LFN* (one dimensional draft).

This problem is solved by adding the input images to the geometric model a second time. This time the viewing rays will be intersected with the LFN in reversed order. The viewing ray number six will not be saved because it was saved in the first run. When viewing ray number five is processed, the angular closest ray is viewing ray number six. Therefore, the colour dissimilarity is detected and the viewing ray is saved by the LFN. The following viewing rays are not saved because of their colour similarity to the angular closest rays. Figure 6.6 (c) shows the saved colour values after the second run.

Experience has shown that it suffices to add the input images three times in alternating reversed order, as there are very few viewing rays saved after the third run.

6.2.3 Rendering

After the colour information has been added to the model, all views are rendered for the display. This is done by ray intersection of all viewing rays with the geometric model. At the first intersection of a viewing ray with a LFN, the intersection test is cancelled, and the colour value of the viewing ray is interpolated by that LFN. The colour is interpolated from angular closest views, as described earlier in Section 6.2.2.

The intersection of viewing rays with the geometric model is a central part of the presented algorithm that is used by all of the different stages. Therefore it is worthwhile to pay particular attention to the parallelization

and implementation of the ray cast. The data structures have been chosen because of their hierarchical partition of space that allow for efficient ray intersection and look-up of the angular closest viewing rays. Another benefit is the inherent ability to handle sparse data sets.

The interested reader is referred to Appendix D for implementation issues.

6.3 Evaluation

The interpolation algorithm was evaluated on foreground and background parts of the scene. This evaluation was performed on the full data set, with about 0.5 percent of the viewpoints selected as reference views. The reference images, the interpolated images, and the difference images between the original and the interpolated images are shown in Figure 6.7. The numbers in the lower right of the different views are used to identify the discrete viewpoints of the display in relation to the nearest reference views, e.g. between the reference viewpoints 00 and 89. The 88 viewpoints that are in between were not used as input images for the interpolation.

The nearest neighbouring reference images of the interpolation for the background are shown in Figure 6.7 (top row), and the nearest neighbouring reference images of the foreground are depicted in Figure 6.7 (second row). For the foreground area, the reference viewpoints are close together to avoid subsampling of the geometry. Another reason is that more light field samples are needed in the foreground area to capture the specular reflections of the golden ornamentation of the mask.

The third row of grouped images in Figure 6.7 shows the results of the interpolation. For each grouping, the top row shows the ground truth input images, as rendered by the modelling tool. In the centre of the figures are the interpolated views, and at the bottom are the negated difference images of the ground truth images and the interpolated ones.

The difference images show that most interpolation errors are located at object boundaries. This is because at object boundaries the depth images either belong to the foreground or the background, and no anti-aliasing can be applied. On the contrary, the colour images were rendered with anti-aliasing, which leads to a colour transfer from the foreground objects to the background and vice versa. Another reason lies in the quantisation of the octree in 3D space, which can lead to colour blending during angular

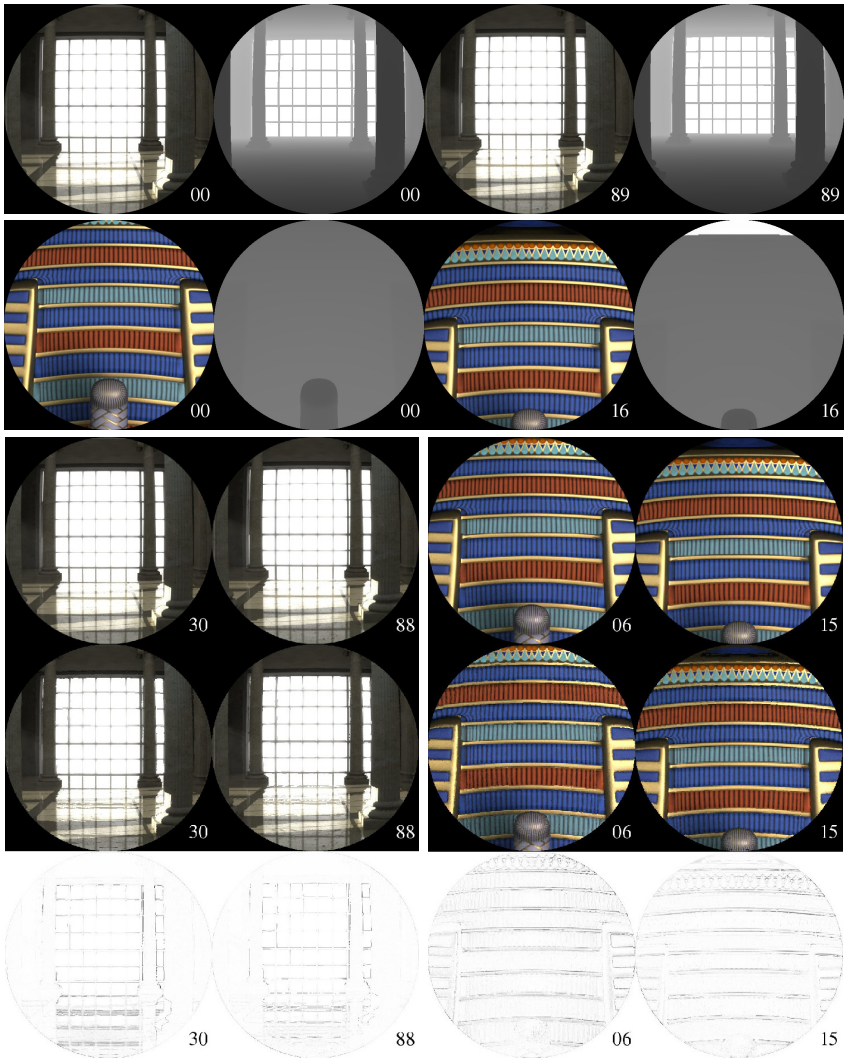

Figure 6.7. The input images are shown in the first two rows. The grouped images in the third row show ground truth on top, the interpolated images below, and the negated difference image at the bottom for the background (left side), and the foreground (right side).

view interpolation on textured surfaces.

6.3.1 Data Reduction

In order to measure the achievable compression rate, the spatial resolution of the input images for the interpolation of the *Tutankhamun* data set was gradually reduced. The results are summarised in Table 6.1.

Table 6.1. Achievable data compression with the introduced approach in the *Tutankhamun* scene. The full data set of 640 x 360 elemental images was interpolated from the spatial reduced input, using approximately every 14^{th} row and column.

Ray tracing	Number of images	[GB]	Percent
All images	230,400	205	100
Colour input images	1,232	1.10	0.54
Depth input images	1,232	0.74	0.36
Summarised input data		1.84	0.90

The amount of input data could be reduced to about one percent, therefore, reducing the initial 205 GB to a total of 1.84 GB, which already includes the depth images.

6.3.2 Runtime

The proposed algorithm has been evaluated on an Intel Core i7-950 CPU with a Nvidia GeForce GTX 460 graphics card. The different implementations and parallelization approaches have been compared for the initialisation and the rendering phase (see Table 6.2). The initialisation phase consists of loading the geometric model, and assembling of the light field. The runtime has been measured on a data set with a spatial reduced resolution, where every 4^{th} row and column has been selected. During the rendering phase, all images of the display are interpolated from a subset of about $1,000$ input views. The results are listed in Table 6.2.

Table 6.2. Comparison of the runtime for the different implementations and the average single image rendering time. The example scene has about 380,000 *LFNs* and 1,000 out of 14,400 input images were used for the interpolation.

Ray tracing	**Initialisation**		**Rendering**		Acc.
	[s]	Acc.	[s]	Acc.	parallel.
CPU (8 Threads)	-	-	84.006	1.00	-
Interpolation					
CPU (1 Thread)	8,610.00	1.00	1.722	48.78	1.00
CPU (8 Threads)	2,041.00	4.22	0.403	208.45	4.27
GPU	750.00	11.48	0.210	400.03	8.20

Ray tracing of an image for the display takes on average 84 seconds, when rendered with the modelling tool. When only the image rendering times are compared, single threaded interpolation already achieves an acceleration factor of 48.78 compared to ray tracing of an image. The parallelization on the CPU using Open Multi-Processing (OpenMP) further accelerates the rendering by a factor of 4.27, yielding an overall acceleration factor of 208. The implementation on the graphics hardware achieved an acceleration factor of 8.20, compared to rendering with a single thread only. This relatively small acceleration factor is because the graphics hardware takes no advantage of the hierarchical data structure of the octree. Nevertheless, the parallelization on the GPU achieved an overall acceleration factor of 400, when compared to the rendering time of the ray tracer.

Figure 6.8 shows the mean rendering times for an image in dependence of the number of LFNs. The results have been retrieved on an Intel Core i7-920 CPU with a Nvidia GeForce GTX 285 graphics card. The straight line in Figure 6.8 was fitted to the data points by a least-squares method. It can be seen that the rendering time has a linear correlation to the number of LFNs that represent the scene. This is because all vertices, which represent the LFNs, are processed by the graphics hardware, and visibility is determined by comparing the depth values. Hence, the graphics hardware does not take advantage of the hierarchical data structure and an early abort for viewing rays is not possible.

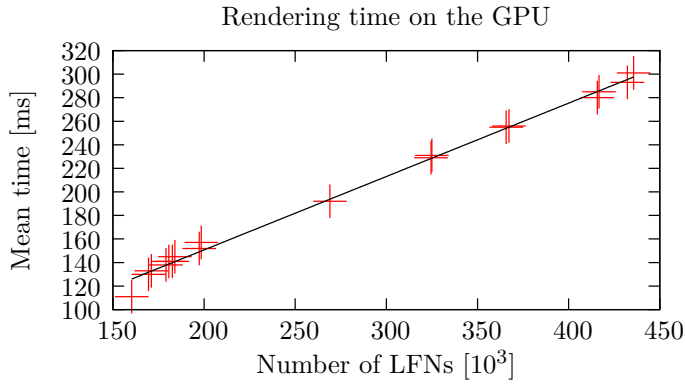

Figure 6.8. Rendering time in dependence of the number of *LFNs* on the graphics hardware.

6.4 Analysis and Conclusion

In this chapter a DIBR algorithm was presented that incorporates a scene representation. It has been shown that the introduced algorithm can be accelerated by parallel hardware, benefiting from both multi-threaded CPU and GPU, available in modern work stations.

With the presented approach, the size of the data set could be reduced to about one percent, reducing rendering time, bandwidth, and storage requirements accordingly. Due to the efficient implementation, rendering could be accelerated by a factor of 400, when compared to the ray tracer that rendered the input images.

However, the results of Chapter 4 and Chapter 5 showed that, depending on the scene, artefacts may occur when the number of input images is reduced. These artefacts are completely scene dependent and it is not feasible to select the number of input images manually. Moreover, by selecting a fixed threshold for the number of input images and using a fixed pattern of spatial reduced input images, it is likely that there are areas where not enough input images are provided, leading to interpolation artefacts, as well as areas where an unnecessary large amount of input images are provided, resulting in wasted bandwidth, computational resources, and storage capacity. This leads to the task to determine the input images in

an automated fashion, a task closely related to viewpoint planning and the BNV problem.

Chapter 7

Scene Analysis for Best-Next-View Selection

The content of this chapter has been previously published in Jung and Koch [JK11a], and Jung and Koch [JK11b]. In case of differences, this publication shall precede the earlier publications.

The previous three chapters have shown that the selection of proper input images for DIBR is a vital task. Depending of the actual scene, the absence of important input images may cause severe artefacts during interpolation. This leads to the challenge of viewpoint selection and viewpoint planning. The problem of selecting the position for the next input image in order to sample a scene, given the current position, is known as the BNV problem.

7.1 An Introduction to the Best-Next-View Problem

This chapter is built on the DIBR algorithm introduced in the previous chapter and is dedicated to the problem of finding an optimal set of input images for the interpolation. In the context of the BNV problem the terms view, and viewpoint are used to describe an elemental image of the display. Establishing a viewpoint ranking of all input images has a number of benefits.

The effects of occlusions have been introduced in Section 2.9.2 and it was shown that occlusions can be arbitrary complex, ranging from different objects occluding each other to self occlusions of concave objects. Another source of severe interpolation artefacts may occur by subsampling of the scene, see Section 2.9.3. The foremost benefit is that a scene analysis allows to interpolate arbitrary scenes because the computation of visibility and completeness of the scene's geometry allows to handle both sources of

interpolation artefacts.

Another advantage is that a scene analysis can be used to discard potential input views that are not important to the interpolation process. Hence, the number of input images can be reduced without jeopardizing the quality of the interpolation.

Finally, a viewpoint ranking allows to balance the costs of additional input images against an expected quality gain of the interpolation, allowing to find a scene dependent number of input images, guided by a quality policy.

7.1.1 Related Work

Early work in viewpoint planning was focused on the optimum position for feature detectability. Tarabanis et al. [TAT95] gives an overview of viewpoint planning algorithms and noticed that the reviewed approaches can be classified into two paradigms they called the generate-and-test paradigm and the synthesis paradigm.

The generate-and-test approach evaluates possible next views and decides on the basis of specific criteria, which view is best suited. In order to reduce the number of possible viewpoints, the viewing space is discretised.

The synthesis approach puts constraints on the BNV that are modelled through an analytic function. The benefit of this approach is that all derived camera positions satisfy the given constraints.

Werner et al. [WHLP96] introduced an algorithm to select the optimum set of reference views out of a set of possible views. In order to solve the problem with dynamic programming, they limited the problem to one degree of freedom.

Massios and Fisher [MF98] were the first to introduce a BNV selection algorithm that uses a quality criterion in addition to the visibility criterion to improve the quality of surfaces. Most BNV algorithms use surface normals to measure the quality of a captured surface or in guiding the decision for the BNV (see Wong et al. [WDA99]).

Best-next-view problems find applications in many real world scanning scenarios, ranging from large scale urban model acquisition (see Teller [Tel98]) to automated object reconstruction (see Levoy et al. [LPC+00]). See Scott et al. [SRR03] for a comprehensive review of automated object reconstruction algorithms. Other applications are automatic camera placement in synthetic (see Fleishman et al. [FCOL99]) and real world (see Byers

et al. [BDG+03]) scenarios. A comprehensive overview and introduction to view planning can be found in Low [Low06].

A lot of research has been done in the context of DIBR that covers the number of samples needed for IBR, their viewpoint (see Schirmacher et al. [SHS99], Lin and Shum [LS00], Zhang and Chen [ZC03a]), and the sampling rate (Chai et al. [CTCS00], Zhang and Chen [ZC03b]). More recent work by Chen and Schonfeld [CS09] extends the prior work about plenoptic sampling to unrestricted parallel cameras and unstructured camera systems and derives necessary and sufficient conditions for unoccluded imaging. Regarding multi-view displays, research has focused on proper sampling and aliasing artefacts (see Halle [Hal94] and Zwicker et al. [ZMD+06]).

The goal of this work is to find an optimum set of input images for a DIBR algorithm that will be used to interpolate views for a multi-view display from a synthetic scene. The presented BNV algorithm for multi-view displays follows the generate-and-test paradigm, where possible viewpoints are defined by the dimension of the display and are discretised by the lens systems of the display. Similar to prior work (see Grossmann and Dally [GD98], Rusinkiewicz and Levoy [RL00], Alexa et al. [ABCO+01], and Corrêa et al. [CFS02]), a 3D point cloud is used to represent the model of the scene.

7.1.2 Input Data and Preconditions

In the following, it is assumed that the depth is known for all viewing rays of the multi-view display. Based on the complete model of the scene, all possible positions shall be sorted according to their contribution to the geometry of the scene and their potential importance for a DIBR algorithm.

To avoid sampling problems and to be close in angle to the interpolated rays, the area of input images shall be restricted to the virtual display plane and the positions of the elemental images of the multi-view display. The angular closeness is of particular importance for the preservation of highlights. Hence, the goal of this algorithm is to provide more samples near the virtual display plane where the opening angle, under which the display is seen, is large.

7.1.3 Integration

The proposed BNV selection algorithm is sketched in Figure 7.1 and is integrated in the DIBR of Chapter 6 as follows. In the first stage the geometric model of the scene is built, using depth images of the scene. After the depth images are added to the scene, the geometric model is refined. In the second stage, all viewpoints of a discrete viewpoint area are evaluated for their suitability as reference images for the proposed DIBR algorithm, incorporating criteria like the size of the display and the distance to the virtual display plane.

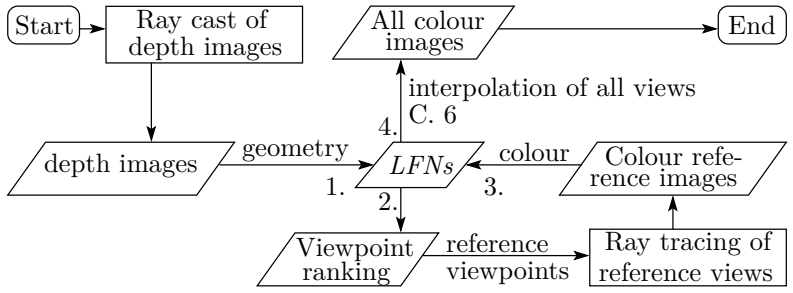

Figure 7.1. Overview of the proposed *BNV* selection algorithm.

The BNV selection is driven by evaluating the visible surfaces of the scene and a ranking of the most important reference elemental images is obtained. These images are fully ray traced and view-dependent surface colour is added as a quad tree for each LFN.

In the third stage, the colour information of the light field is assembled from the previously chosen set of input images. The final stage is the rendering of all views for the full parallax display.

7.1.4 Best-Next-View Selection

This section gives an overview of the design principles and display dependent design choices that will influence the proposed BNV selection algorithm. The depth images for each elemental image, which are required by the presented approach, can be computed much faster than the ray traced colour images, at marginal costs. The possible views are restricted by the dimension and

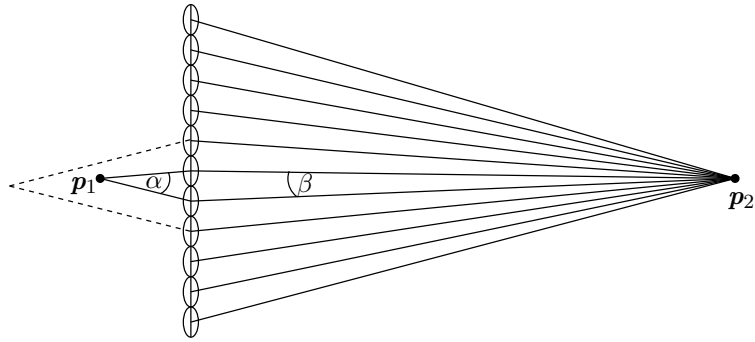

Figure 7.2. Scene sampling in relation to the distance to the virtual display plane.

placement of the display in relation to the scene, in accordance to the generate-and-test approach.

Full parallax displays emit a light field in a display dependent FOV. The viewing frustum is defined by the FOV of the elemental images and the viewing frustum of the multi-view display is the union of the viewing frustum of all its elemental images (cf. Section 2.3.1 and Figure 2.6).

If all images of the display are rendered, the displayed scene is sampled in dependence of its distance to the display plane, due to the limited field of view of the single elemental image. The relation is depicted in Figure 7.2. The viewing rays with the maximum angle of incidence are drawn dashed for the closest picture elements that can not sample $\boldsymbol{p}_1$. Point $\boldsymbol{p}_1$ is close to the virtual display plane and can only be sampled by two elemental images with a difference in angle of α, whereas the point $\boldsymbol{p}_2$ is further away from the virtual display plane and is sampled by twelve elemental images with a relatively small difference in angle β, between neighbouring elemental images.

The consequence for view point planning is that, in general, content near the display plane can only be sampled by few views, hence, restricting the number of potential input views, whereas content far from the display plane can be sampled by a large number of viewpoints with little impact to the fidelity of the sampling.

Therefore, the distance of a point in the scene to the display plane will be

used to measure its importance for the interpolation, and will determine the ranking of the associated input views, which affects the number of samples available for the view interpolation. The relation between the distance d_z to the virtual display plane, the displays diagonal $\| S_v \|_2$, and the viewing angle α is given by

$$d_z = \frac{\| S_v \|_2}{2 \cdot tan\,(\alpha/2)} \tag{7.1.1}$$

for point $\boldsymbol{p}_1$ at the centre of the display and the two dimensional case. The relation is also depicted in Figure 7.3. The further the 3D point is away from the display, the smaller is the difference between the possible angles, under which the point can be observed by the display.

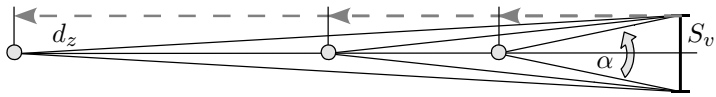

Figure 7.3. The difference in the viewing angle in relation to the distance of the display and its diagonal size.

Hence the scene can be approximated as ambient at greater distances, because the view dependent shading part does not change significantly for the different views of the display. Therefore points near the virtual display plane will be regarded as potential important while points far away from the display plane will only be of geometrical importance, i.e. it is sufficient to sample them once.

7.2 Algorithm

A typical scene of the 3D poster display consists of about 400,000 3D points and 250,000 colour images for the lens systems of the multi-view display, with 50,000 viewing rays per image. Therefore, finding the optimum input images for a DIBR algorithm by exhaustive search is not feasible. The goal is to find a solution that is close to the optimum and suffice several conditions.

7.2.1 Optimisation

First, the geometry should be as complete as possible to avoid holes that will lead to artefacts during reconstruction of the light field. Another requirement is that foreground objects should have more weight when considering input images than background objects, because view dependent changes are more likely to occur near the virtual display plane. On the other hand, it is important to have a measurement about how many input images are at least needed, and how an additional input image effects the result of the image interpolation. Hence, the importance of a viewpoint is weighted by the number of visible 3D points and its viewing distance to each 3D point.

The viewpoints are restricted to the positions of the elemental images, following the generate-and-test paradigm as introduced by Tarabanis et al. [TAT95].

Geometric Model

The scene is represented by a 3D point cloud that is inserted into an octree. All viewing rays of the display, i.e. all viewing rays of all elemental images, are cast and the depth information is used to generate the 3D point representation of the scene. Numerical precision errors in the depth information will generate points lying under the surface, which are removed at the end.

Visibility and Completeness of the Model

After the geometric model has been build, the visibility is computed. There are two strategies to compute the visibility. The first is to place a virtual camera at every 3D point of the geometric model, oriented orthogonal to the display plane, and project the display. This has the drawback that the resolution of the virtual camera depends on the distance to the virtual display plane. Therefore, the chosen strategy used for the presented algorithm is to cast a ray for every viewing ray of the display and intersect it with the geometric model.

For every 3D point, the position of the elemental images that intersect the LFN is saved in memory. Due to memory limitations it is not possible to save all image positions for every 3D point for large scenes and large displays with many elemental images. Therefore only five image positions are saved per 3D point, the image positions with the minimum and maximum angle on

the horizontal, and vertical axis of the display, and the image position that is nearest to the centre of the opening angle. These positions will be used during the optimisation to sort the positions according to their importance.

Even simple scenes can have occluded regions that are only visible from a single viewpoint. Two cases are depicted in Figure 2.21. On the left side an object is occluded by another foreground object in such a way that the dashed drawn view cannot be interpolated from the neighbouring view without artefacts. The right side of Figure 2.21 shows a concave object with self occlusion, and again, the dashed drawn viewpoint cannot be interpolated from the neighbouring view without artefacts.

Another source of interpolation artefacts is subsampling of the scene by elemental input views that are too far apart. This situation is depicted in Fig. 2.22 on the left side. In the dashed drawn viewpoint the object cannot be interpolated from the neighbouring views, leading to a hole in the object through that the background can be observed.

The presented BNV algorithm depends on a complete geometric model of the scene. By registering the geometric model against all viewpoints, all kinds of possible occlusions and subsampling are implicitly handled.

Initialisation

The introduced optimisation will be iterative and the number of iterations, until the algorithm converges, depends on the quality of the starting value. Figure 7.4 gives an overview of the proposed BNV algorithm that can be separated into an initialisation phase and the iterative optimisation of the viewpoints.

First, the geometric model is built and the surface is extracted. Then the number of unique 3D points are counted for every image position, i.e. the number of 3D points that are seen only by that image. If these images are omitted, the geometric model for the DIBR algorithm will be incomplete, which may result in visible holes. These unique viewpoints are stored on a special list l_u that is only computed once. The viewpoints on list l_u are always the first input images that are added and are excluded from the viewpoint optimisation. Then, the positions of the five elemental images are evaluated, which are saved by every 3D point. The saved image positions are summed up and weighted with the distance of the 3D point to the virtual display plane.

Figure 7.3 shows the relation between the distance d_z to the virtual

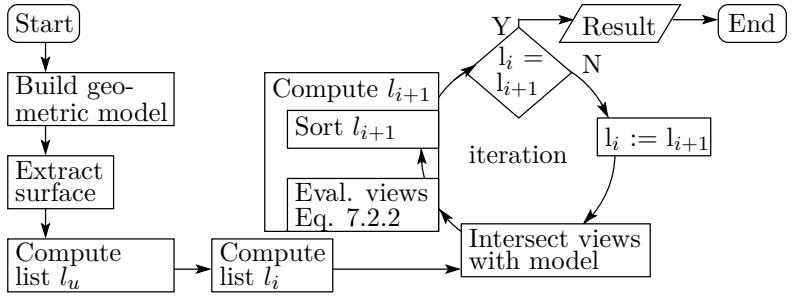

Figure 7.4. Schematic overview of the proposed algorithm.

display plane, the diagonal size $\| S_v \|_2$ of the display, and the viewing angle α for a point $\boldsymbol{p}$, located on a line that runs orthogonally through the centre of the display. Solving Equation 7.1.1 for the viewing angle, the relation between the viewing angle and the distance to the virtual display plane is described by

$$\alpha = 2 \cdot \arctan \frac{\| S_v \|_2}{2 \cdot d_z} \ . \tag{7.2.1}$$

It can be seen that the opening angle becomes smaller, as the distance to the virtual display plane increases, therefore, points far away from the virtual display plane can be safely regarded as ambient and need to be sampled less often than points near the virtual display plane, which may contain view dependent colour information as they are sampled under a large opening angle from the display.

To account for this, each visible 3D point is weighted by its squared Euclidean distance to the virtual display plane. A global list l_i is created where all possible viewpoints are ordered by their measured importance. This allows for an interactive selection of the number of input images that are used as input images for the DIBR algorithm.

Let $N = R_{vX} \cdot R_{vY}$ be the number of elemental images of a multi-view display, $\boldsymbol{V}_n$ the position of the lens system with $n \in \{1, ..., N\}$, M be the number of 3D points that define the geometric model, and $\boldsymbol{p}_m$ the position of a 3D point with $m \in \{1, ..., M\}$. The importance w_{I_n} of an elemental input

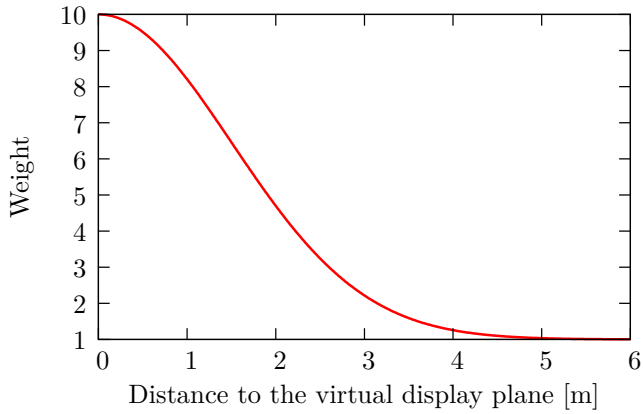

Figure 7.5. Plot of the weighting function with $\sigma = 1.5$, $w_F = 9.0$, and $w_M = 1$, for distances up to 6 meters.

image I_n is then measured by the sum over the weights of all 3D points as

$$w_{I_n} := \sum_{m=1}^{M} w_m(\boldsymbol{V}_n, \boldsymbol{p}_m, \sigma), \tag{7.2.2}$$

with the display fitting value σ, a scaling factor w_F, and a minimum weight w_M. The weighting function w_m is defined as

$$w_m(\boldsymbol{V}, \boldsymbol{p}, \sigma) := \begin{cases} 0, & \text{if } \boldsymbol{p} \text{ is not visible from } \boldsymbol{V} \\ w_F \cdot e^{-\|\boldsymbol{p}-\boldsymbol{V}\|_2^2/(2\sigma^2)} + w_M, & \text{else.} \end{cases} \tag{7.2.3}$$

Figure 7.5 shows an example of the weighting function for a display with about 1.5 meter display diagonal. The weighting function can be adjusted to the display size and the angular resolution of the multi-view display by adjusting the display fitting value σ. An exponential decay function was used to ensure that the large number of 3D background points do not dominate the optimisation process. For perceived quality the fidelity of the foreground objects is more important and was chosen to dominate the optimisation.

This weighting will favour images that sample objects near the display

plane over background images. The level, that images close to the virtual display plane are favoured, is controlled by the factor w_F and the minimum weight is controlled by the parameter w_M. The mapping of each image to the number of 3D points is used to generate a list of the images, sorted by the number of 3D points that are attached to an image. This list is used as starting point for the optimisation.

Iterative Optimisation

Every 3D point has a boolean attached to it, marking it as active or inactive. At the beginning of an iteration all 3D points are set inactive. Then, the unique viewpoints of the l_u are processed. For every viewing ray of an image a ray is cast and intersected with the geometric model. If a 3D point is hit and inactive, it will be marked as active, and it will be related to the elemental image that hits it first. Next, the remaining list l_i is processed, starting with the first entry of the list. The camera is placed at the position of the first elemental image from the list and all viewing rays are cast. After all input images are processed, or all 3D points are set active, a new global list l_{i+1} is generated by evaluating all views according to Equation 7.2.2, and sorted by the number of 3D points that are attached to each image. This list is used as input sorting for the next iteration.

Stop Criterion

When the new list l_{i+1} is identical with the current list l_i, the optimisation is discontinued. The evaluation of one iteration can be very time consuming for large displays and complex scenes. Therefore the number of changes to the list can be used as an alternative stop criterion.

7.2.2 Properties and Limitations

The optimisation depends heavily on the initialisation. It is very unlikely that images that are once marked dispensable will ever be considered as input images, although, they might be part of a better solution. The optimisation will be executed mainly on the elemental images found during computation of the visibility. Therefore, it can be used to guide the optimisation to mainly be executed on positions that are seen as favourable for DIBR algorithms.

The output is the global list of all possible viewpoints, arranged in descending order by the importance as input image for DIBR, that allows to successively add input image to the interpolation. In addition, the overall geometric completeness of the model and the contribution of a single input image are available, allowing to determine the number and position of the input images before the first colour image is rendered by a ray tracer.

7.3 Evaluation

Figure 7.6 (left) shows an overview of the synthetic scene that was used for evaluation. For the BNV selection algorithm, it is sufficient to use a spatial reduced resolution of the data set, where every fourth row and column of the full data set is selected. Figure 7.6 (left) was created by a simulation of this reduced display where the viewpoint was centred about 60 centimetres in front of the display (cf. Section 3.2).

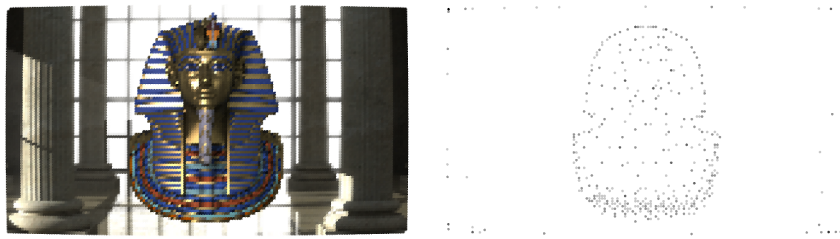

Figure 7.6. Simulated display overview with the viewpoint centred about 60 centimetres in front of the display (left) and position of the top ranking 393 viewpoints (right). The darker a viewpoint is coloured, the more important it is ranked.

7.3.1 Geometric Completeness

The reduced data set consists of 14,400 elemental images with a resolution of 512x512 pixels. Figure 7.7 (left) shows the contribution of each input image to the geometry of the scene where only newly inserted 3D points are counted. The first ranking viewpoint adds about nine percent of the

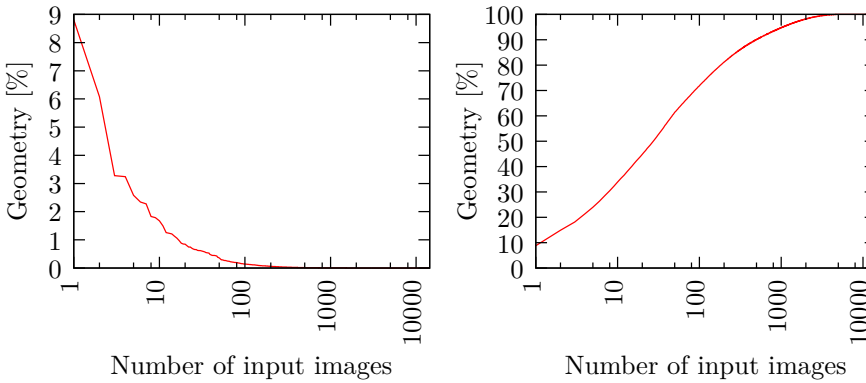

Figure 7.7. Contribution of the single input images to the geometry of the model (left) and overall completeness of the model in dependence of the number of input images (right). The axes of abscissae are in logarithmic scale.

geometry to the model. Around the 100^{th} input image, the contribution of new 3D points to the model's geometry is only marginal.

In Figure 7.7 (right) the overall completeness of the model is shown by integration over the contribution of the images. It can be seen that after the first 100 images are added, the obtained geometry is incomplete and covers only 70 percent of the scene. This is due to the many marginal contributions of single images that add up to about 30 percent of the complete geometry. For the complete geometry, the first 4,762 input images are required, which is about 33 percent of the input image set. However, even with only 1,000 reference images, which is about 7 percent of the reduced image set, no visible artefacts can be seen and a high quality reconstruction can be achieved.

The final result of the BNV algorithm is pictured in Figure 7.6 (right). The perspective is the same as in the left figure. Therefore, the selected viewpoints can be directly related to the overview. The importance of a viewpoint is represented by its brightness with the most important viewpoint set to black. Viewpoints with a minor contribution to the scene's geometry have been coloured white for the sake of clarity. The two viewpoints that are rated most important are in the upper left corner of the display, followed by the lower right corner of the display. Both viewpoints cover the background,

which has a larger surface than the foreground. Therefore, the majority of the 3D points, that are required to describe the geometry of the scene, belong to the background, which makes those viewpoints most important for the geometric completeness. The second rated viewpoint lies in the opposite display corner of the most important viewpoint, where most of the background regions could be sampled, which were occluded in the first viewpoint.

In general, it can be seen that the foreground on the mask is sampled more often than the background of the scene. This is because the weighting function is set to prefer foreground objects as they are more relevant for view dependent colour information. Another reason is that near the virtual display plane, sampling points must be closer for an overlapping FOV or holes in foreground objects may occur.

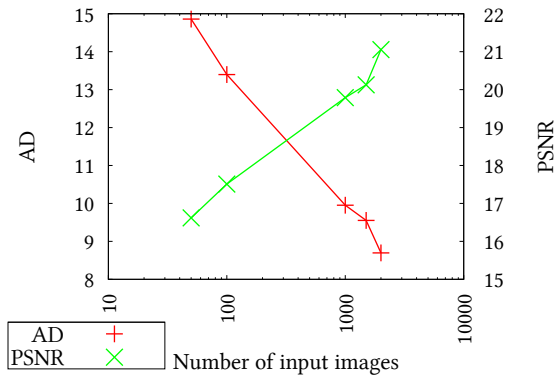

Figure 7.8. *AD* and the *PSNR* of the simulated overview (reduced data set) from the reconstructed images, compared to the simulated overview from the ground truth images. The axis of abscissae is in logarithmic scale.

The results are depicted in Figure 7.8. The evaluation has been performed on the top ranking 50, 100, 1,000, 1,500, and 2,000 images from the BNV algorithm. As expected, the AD is drastically reduced when the number of input images is increased. In the reconstructed images, the object boundaries are one to two pixels off due to aliasing, which results in a relatively low PSNR.

Figure 7.9 shows close up views of the foreground area, comparing the

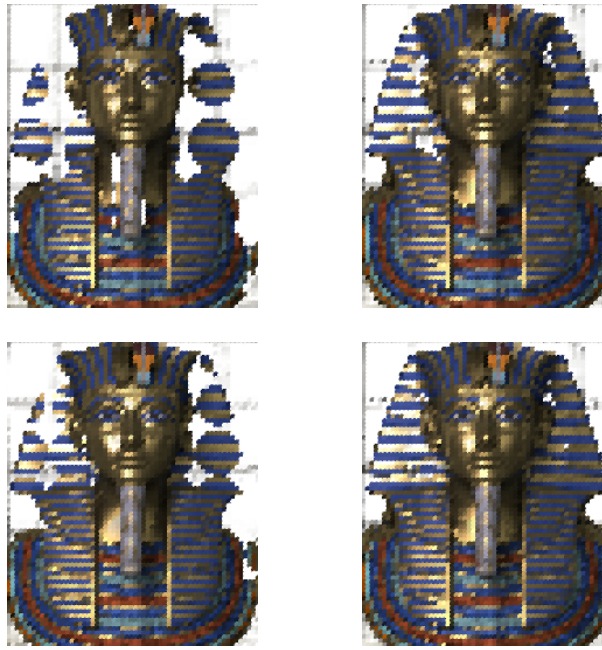

Figure 7.9. Close up of the foreground area of a simulated display overview (reduced data set) from reconstructed images using 104 input images (top row) and 144 input images (bottom row). Comparison of regular sampled viewpoints (l) against the same number of top ranking viewpoints from the *BNV* algorithm (r).

simulated display overview from reconstructed images with the optimised input images against a simulated overview from the reconstructed images with regular sampled input images, using 104 input images (top row) and 144 input images (bottom row). In each case, the number of input images was the same and all images of the reduced data set were interpolated with the same DIBR algorithm. The results on the left side were obtained by using regular sampled viewpoints for the interpolation and the right result was obtained by using the same number of top ranking viewpoints from the BNV algorithm for the interpolation.

In Figure 7.9 it can be seen that with regular sampling the reconstruction

with 104 and 144 input images lacks parts of the geometry near the virtual display plane, due to non-overlapping fields of view and occlusions, leading to severe artefacts in the reconstruction. When the top ranking viewpoints were used for image reconstruction, the geometry of the foreground is much more complete. Although there are still holes in the geometry of the foreground object, the geometry of the foreground is almost complete using 144 input images, resulting in less artefacts compared to the reconstruction from regular sampled viewpoints.

Good image interpolation results can be achieved with less than 100 percent of complete geometry and without manual user interaction by the proposed solution. For premium high-quality rendering, additional reference views can be progressively added from the ranking, which has been obtained by the BNV selection algorithm. The result can be interactively evaluated on the geometric model and further reference views can be added until a satisfying result is obtained.

7.3.2 Distribution of the Light Field

In this part of the evaluation, the number of relevant light field information per input image is investigated. A relevant light field sample is defined as a pixel of an input image that is saved during assembly of the light field (see Section 6.2.2). For a Lambertian surface, only one light field sample will be stored, therefore, for LFNs representing a Lambertian surface, only the first input image can contribute to the stored light field.

The plot of the relevant light field information per input image (see Figure 7.10, left) is very sharp declining. This is due to two images that have a very high rating, covering a large portion of the background. The progress of the plot shows that a lot of input images have a marginal contribution to the light field, compared to the total amount of samples. The conclusion is that most surfaces of the scene expose Lambertian properties or only marginal variation in the colour.

Because of the higher number of light field samples per 3D point, the number of relevant positions is expected to be much higher for the light field information compared to the number of positions for the geometric model. This is supported by Figure 7.10 (right), which shows the overall completeness of the light field information with respect to the maximum number that is stored in the data structure.

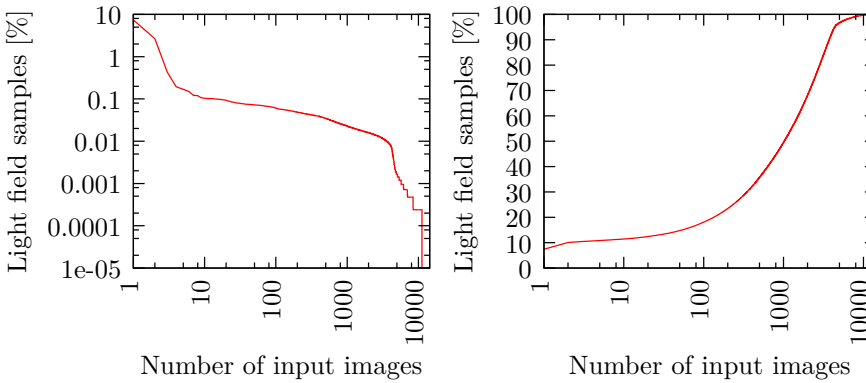

Figure 7.10. Percent of the light field information per input image (left), and overall completeness of the light field information with regard to the maximum number of samples that are stored for this scene (right). In the left figure, both axes are in logarithmic scale, in the right figure only the axis of abscissae is in logarithmic scale.

7.3.3 Runtime

The proposed algorithm has been evaluated on an Intel Core i7-950 CPU with a Nvidia GeForce GTX 460 graphics card. The different implementations and parallelization approaches have been compared for the initialisation and the rendering phase. The initialisation phase consists of loading the geometric model and assembling of the light field.

The resulting acceleration for the reduced data set of the last chapter, which are given in Table 6.2, do not take into account the rendering time of the input images and the viewpoint selection. Table 7.1 shows the effective acceleration that can be achieved for a full sized display with 230,400 lens systems. The top ranking 1,000 input images of the BNV selection algorithm were chosen for the interpolation, the same number of onput images as in Table 6.2. The upper part of the table shows the rendering times when all images for the display are ray traced and computed on a single work station.

The centre of the table shows the different stages that are required for the interpolation. First, the depth images are ray traced with the modelling tool. For the BNV selection algorithm, it is sufficient to sample only every

Table 7.1. Effective acceleration of the rendering compared to ray tracing (RT) of all colour images for the full data set.

	Rendering method	Number of images	Time [h]	Accel.
Ray tracing	RT_I (8 Threads)	230,400 colour	5,376.38	1.00
Initialisation	RT_D (8 Threads)	14,400 depth	4.00	-
	GPU (*BNV*)	-	5.30	-
	RT_I (8 Threads)	1,000 colour	23.34	-
View interpolation	GPU_R (rendering)	230,400 colour	22.19	242.29
Effective acceleration	$\sum RT_D$, *BNV*, RT_I, GPU_R	230,400 colour	54.83	98.06

fourth row and column of the data set. As long as parts of the scene do not lie inside, or extremely close to, the virtual display plane, the reduced data set contains enough geometric detail for proper viewpoint selection. The viewpoints are ordered by the BNV selection algorithm, followed by ray tracing of the chosen viewpoints with the modelling tool. Finally, the interpolation is initialised and all images of the display are interpolated. For interpolation, the algorithm implemented on the graphics hardware was chosen.

The lower part of Table 7.1 shows the sum of all steps involved, including the rendering time of the input colour and the depth images. It can be seen that the acceleration factor is mainly influenced by the ratio of the interpolated images to the input images that are required for interpolation. For a full sized display, an effective acceleration factor of almost 100 was achieved by the proposed algorithm.

7.4 Conclusion

Finding the optimum set of input images for a DIBR algorithm is a difficult task, because it solely depends on the modelled scene. An important

precondition before DIBR can be integrated into the production of content for 3D poster displays is to ensure that prominent artefacts are not introduced by the interpolation process. This is especially the case for missing geometry, caused by subsampling of the scene, or by occlusions.

The presented approach uses the depth images of a display with a reduced spatial resolution to analyse the importance of the viewpoints, building an optimised viewpoint ranking regarding the contribution of an input view to the geometric completeness of the modelled scene. The obtained viewpoint ranking determined the optimal input images that were afterwards rendered with the ray tracer.

It was shown that this approach allows to reduce the amount and severity of interpolation artefacts, when compared to the same amount of input images from a of set of spatial regular subsampled input images. Compared with the computational costs, a ray tracer requires for the rendering of the colour images, the computational resources required by the BNV selection algorithm are negligible. The overall acceleration achieved with the presented approach was about two orders of magnitude, when compared to rendering all images with a ray tracer. This reduces the rendering time of all images of a full parallax poster display from several month to under three days, and offers an opportunity to measure the geometric completeness before initiating the costly rendering process of the ray tracer.

Chapter 8

Interpolation for Multi-View Video Displays

The content of this chapter has been previously published in Jung and Koch [JK13]. In case of differences, this publication shall precede the earlier publications.

The content creation and display properties of full parallax video displays differ from the requirements of full parallax poster displays in several aspects. The foremost difference is that a video display switches the displayed content around 20 times per second, resulting in data rates of up to several hundred GB per second (cf. Section 2.7.2). This data rate challenges the rendering, transmission, and storage of the displayed content. Moreover, the playback of video content imposes a real-time requirement on the interpolation process.

In order to minimise the rendering time, data rate, and storage demands of the displayed content, DIBR will be used to reduce the rendering to a small subset and interpolate the full data set on the device in real-time. The evaluation of Chapter 4 and Chapter 5 has shown that the quality of the interpolation depends heavily on the arrangement of the scene, the number of input images, and the representation of the input images.

As a consequence, the number of input images, and accordingly the required data rate, varies with the arrangement of the scene, when prominent interpolation artefacts shall be avoided. This contradicts the requirement of an interpolation in real-time, because the available computational resources of a video display have to be decided on during the design of the display, resulting in an upper bound of the computational resources, driven by economically interests. With unstructured input data, a worst case scenario has to be assumed to ensure playback without frame drops, requiring to provide a huge amount of computational resources. With a structured input, the number of executed instances is always the same, hence, the required computational resources vary by small amounts only, under the constraints

that the processed data has little impact on the program flow.

Chapter 6 proposed a DIBR algorithm for 3D poster displays that uses hierarchical data structures, maps, and a lossy encoding scheme for the colour information. For the real-time processing of video data this complex approach is unsuitable, because computationally too demanding.

In Chapter 7, the spatial distribution of the most important input images was investigated, amongst other things. Figure 7.6 shows the distribution of these elemental input images for the *Tutankhamun* scene, and it can be seen that the distribution of the top ranking input views heavily depends on the arrangement of the scene, which explains the variations in the fidelity of the interpolation between the different scenes, when the number of input images, the distribution, and the image representation were fixed throughout the Chapters 4 and 5.

It follows that the location of the input data on the display varies with the displayed content. This will influence the memory access pattern of the interpolation process and causes an unequal distribution of computational requirements for approaches that parallelize the interpolation process by spatial partitioning.

In consequence of these requirements, conventional approaches are not well suited to design a DIBR algorithm for a full parallax video display. Summarizing, the requirements for the interpolation of content for full parallax video displays are a data rate and memory access pattern that varies as little as possible to ensure computation in real-time, an interpolation algorithm that is computational inexpensive, a structured location of the input data to allow parallelization by spatial partitioning, a massive reduction of the input images in order to reduce the amount of processed data, and a way to prevent prominent artefacts, caused by occlusion or subsampling of the scene.

The DIBR algorithms introduced in Chapter 4 and Chapter 5 already fulfil the requirements of a computational inexpensive interpolation, because a basic pixel warping algorithm is used, combined with a weighted blending. In addition, the input images for the interpolation of a view have an upper bound of the nearest four neighbours, which allows for a structured input and limits the memory access to a maximum of the nearest four input images.

In order to achieve a structured location of the input images and to reduce the input data rate by a large amount, the effects of very sparse input data, that is evenly distributed in the interpolated domain, are investigated.

8.1 Sparse and Evenly Distributed Input

The evaluation of Chapters 4 and 5 concludes that the quality of the interpolation degrades with a decreasing number of input images. When the input images fall below a scene depending number of input images, prominent artefacts occur due to subsampling of the scene or occlusions.

The noticeable difference between the interpolation of orthographic input images and the interpolation of perspective elemental images was that by decreasing the number of input images, prominent artefacts manifested exclusively on the background in the orthographic case, and exclusively on the foreground in the perspective case.

Another difference that could be noticed by comparing the results of Chapters 4 and 5 was that the PSNR of the orthographic input images degraded in a much flatter slope, when compared to the perspective input images, although the perspective images started on a higher PSNR. This raises the question of how far the orthographic input images can be reduced, while yielding still reasonable interpolation results for foreground objects. Hence, in the following it is evaluated how the reduction to a very small amount of orthographic input images affects the interpolation result, and especially the fidelity of the foreground objects.

Figure 8.1 shows the averaged PSNR of all simulated viewpoints in relation to the number of input views for the *Coffee Capsules* scene on the left, and the *Tutankhamun* scene on the right for orthographic views. For this evaluation, the investigated interval of Chapters 4 and 5 has been expanded, reducing the number of input views down to 0.015 percent. Hence, the axis of abscissae is also plotted in logarithmic scale.

It can be seen, that the relation of the mean image fidelity is almost linear to the number of input images, when the evaluation of the *Tutankhamun* scene with the most input views is ignored. It can also be seen, that the averaged mean, maximum, and minimum PSNR are almost parallel, except for the averaged maximum PSNR of the *Tutankhamun* scene.

Figure 8.2 shows the viewpoint with the minimum PSNR of 23.91 dB (viewpoint C_{35}, top row), the viewpoint with the maximum PSNR of 26.68 dB (viewpoint C_{178}, middle row), and the plots of the PSNR and AD for all evaluated viewpoints (bottom row), with 0.015 percent of all rays as input for the interpolation. The top two rows show the ground truth, the interpolated, and the negated difference images, from left to right.

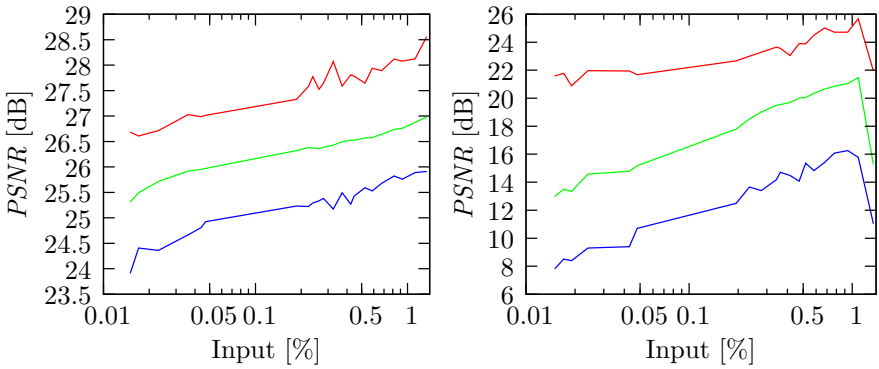

Figure 8.1. Line graph of maximum (top), mean (middle), and minimum (lower) *PSNR* of scene *Coffee Capsules* (left) and scene *Tutankhamun* (right), averaged over all simulated viewpoints and related to the percentage of input viewing rays for orthographic views.

Similar to the evaluation that used 0.28 percent of the input views (cf. Figure 4.4), the relation between the viewpoint and the PSNR has little variation, about 3 dB in the evaluated interval. Although the overall PSNR is relatively high, when compared to the *Tutankhamun* scene, prominent artefacts are observed in the viewpoint with the minimum PSNR (C_{35}, top row). These artefacts are due to occlusions, caused by the large number of floating capsules. In the interpolated viewpoint with the maximum PSNR (C_{178}, middle row), the affected capsules are out of the field of view. In addition, a lot of capsules are occluded in that view, which might favour the interpolation, allowing for an interpolated view without prominent artefacts. The line up of the capsules also reduce the occupied area of the capsules in the image, hence reducing the interpolation error. This is supported by the observation that the viewpoint with the maximum PSNR stayed in the neighbourhood, when compared to the evaluation with 0.28 percent of the input views, whereas the viewpoint with the minimum PSNR has relocated.

Figure 8.3 shows viewpoint Position 64 with the minimum PSNR of 7.81 dB, Position 259 with the maximum PSNR of 21.58 dB, and the plots of the PSNR and AD for all evaluated viewpoints from top to bottom, of the interpolation with 0.015 percent of all rays as input. The top two rows show the ground truth, the interpolated, and the negated difference images,

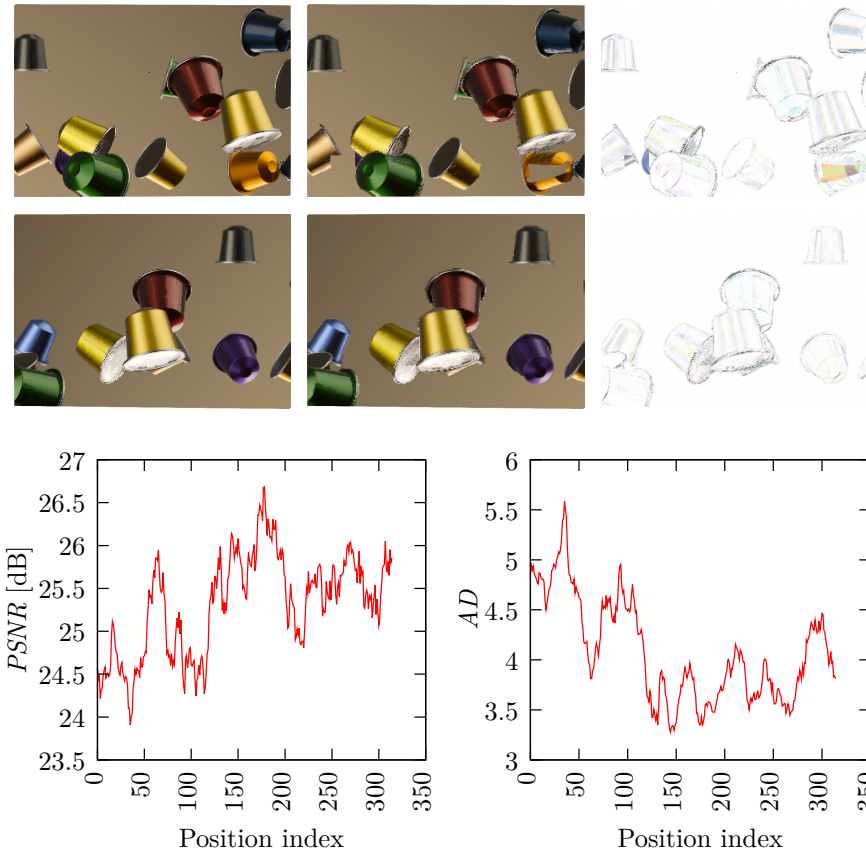

Figure 8.2. Upper row: Simulated view of position C_{35} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{178} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: *PSNR* (left) and *AD* (right) of scene *Coffee Capsules* with 0.015 percent of all rays as input. The axis of abscissae shows the viewpoint (cf. Figure 3.4).

from left to right.

The relation between the viewpoints and the PSNR still has some resemblance to the relation in Figure 4.9 and the PSNR varies in the

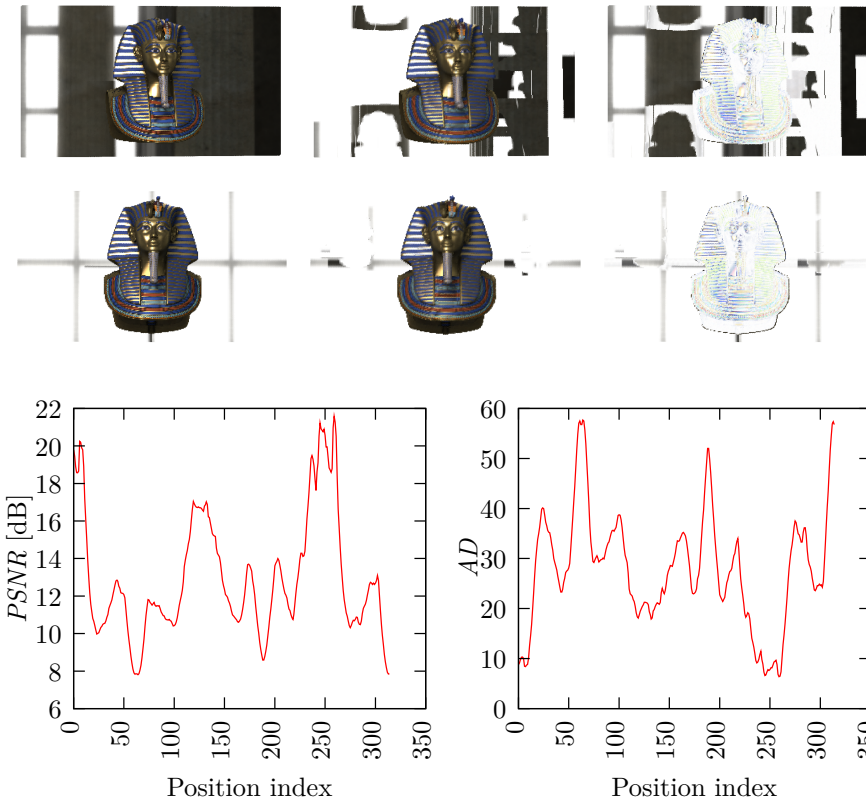

Figure 8.3. Upper row: Simulated view of position C_{64} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{259} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: *PSNR* (left) and *AD* (right) of scene *Tutankhamun* with 0.015 percent of all rays as input. The axis of abscissae shows the viewers position (cf. Figure 3.4).

interval of about 13 dB. The reason for the very shallow descent of the averaged maximum PSNR of the *Tutankhamun* scene in Figure 8.1 can be observed by comparing the interpolated viewpoint with the maximum PSNR (C_{259}) using 0.28 percent of the input images (Figure 4.9), with the

interpolated viewpoint with the maximum PSNR (C_{259}) using 0.015 percent of the input images (Figure 8.3). The viewpoints show the scene from a perspective where the foreground object covers a maximal area in the image, showing even a part of the inner surface, whereas the background covers a very small area in the image, that consists exclusively of the lattice and a constant background colour.

The viewpoints with the minimum PSNR shows the scene from a similar viewpoint, where the structured background covers a very large area of the interpolated images. In contrast to the averaged maximum PSNR, the averaged minimum PSNR of the *Tutankhamun* scene in Figure 8.1 is almost parallel to the averaged mean PSNR, accounting for the majority of the loss in overall image fidelity with the decreasing number of input views. It can be concluded that the background is primary affected by the decrease in image fidelity, caused by reducing the input images, whereas the foreground objects show a little decrease in image fidelity. This is also supported by the negated difference images of the interpolated views.

The evaluation showed that the number of orthographic input images could be reduced to very few images, while still yielding reasonable interpolation results without prominent artefacts on foreground objects. This observation shall be exploited by combining the interpolation results from perspective elemental, and orthographic images, guided by the corresponding depth maps. Content near the virtual display plane will be interpolated from orthographic images, whereas content far away from the virtual display plane will be interpolated from the elemental perspective images.

8.2 Properties of Image Representations

The question remaining is how to switch between the interpolated orthographic and perspective results. In order to find a meaningful distance, the properties of orthographic and perspective images regarding the connection of depth and disparity are reviewed, as well as the sampling properties of the different representations, regarding the scene. The relationship between depth and disparity had already been formally introduced in Section 2.8.3, therefore, at this point an informal summary is given.

8.2.1 Orthographic Images

Figure 8.4 shows the display plane and the viewing frusta of two different orthographic input images. It can be seen that the frusta completely overlap at the display plane, which is sampled by every orthographic input image. Due to the parallel viewing rays, the sampling frequency inside the frusta is

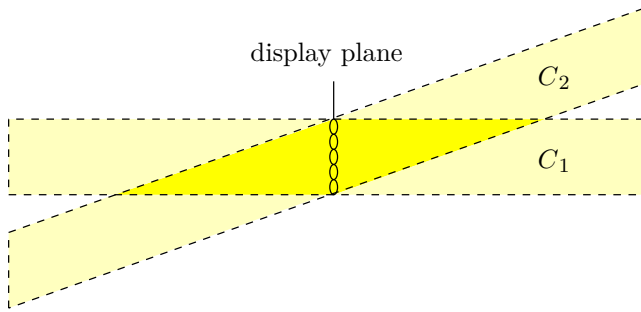

Figure 8.4. Sampling of the scene with orthographic input images. Towards the outer regions, the viewing frusta does not overlap, leading to unseen areas of the scene.

constant, leading to non-overlapping FOVs in the outer regions. At infinity the orthographic views do not overlap, hence, this is the worst case for the interpolation of orthographic images, and an interpolation is not possible.

Figure 8.5 shows selected rays from two orthographic view directions C_1 and C_2, drawn solid and dashed. At the distances d_1 and d_2 along a ray of view C_1, the corresponding disparities δ_1 and δ_2 for view C_2 are plotted. It can be seen that objects in the display plane are sampled at

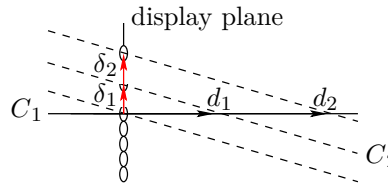

Figure 8.5. Relationship between disparity and depth for orthographic input images.

the same location by all ray directions of a single lens system. Hence, the resulting disparity is zero. With increasing distance to the display plane, the disparity grows, until the border of the display is reached, leaving the viewing frustum of the dashed orthographic view.

8.2.2 Perspective Images

Figure 8.6 shows two perspective elemental input images and the corresponding viewing frusta. It can be seen that the input images have no overlapping FOV near the display plane, but progressively overlap with an increasing distance to the display. The worst case for the interpolation of perspective

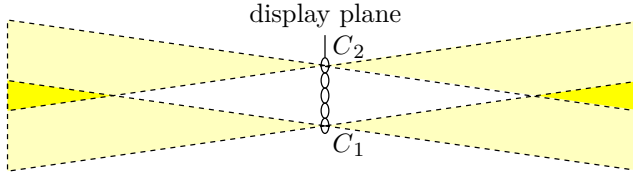

Figure 8.6. Perspective sampling of elemental images of the scene.

images is content in the display plane, where even neighbouring views do not overlap.

Figure 8.7 shows two elemental input images C_1 and C_2. Along the stippled ray of C_1, the intersection with the outer most ray of C_2 is plotted at distance d_1, with the corresponding disparity in C_2 of δ_1, and the intersection with a ray of C_2 at a greater distance d_2, with the corresponding disparity δ_2. For the relationship between depth and disparity follows, that starting

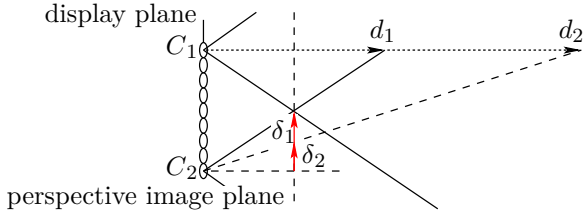

Figure 8.7. Relationship between disparity and depth for perspective elemental input images.

from the maximum disparity at the point the viewing frusta overlap, the disparity becomes smaller with an increasing distance of the object to the display plane, approaching zero at an infinity distance.

8.2.3 Combining the Properties for Interpolation

Figure 8.8 shows a draft of the display plane, which coincides with the orthographic image plane. The perspective image plane in Figure 8.8 is drawn at distance d for clarity. Two elemental perspective images C_{P1} and C_{P2} with baseline $\flat$ are drawn as solid black lines and the two orthographic input images C_{O1} and C_{O2} with an angular difference of α are drawn in cyan.

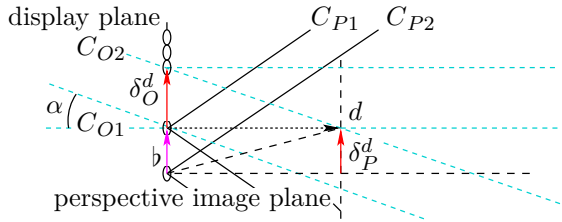

Figure 8.8. Relationship between orthographic and perspective elemental input images, with regards to disparity and depth.

At distance d the corresponding disparity between the orthographic input images is δ_O^d and for the perspective elemental images δ_P^d. For content within distance d to the display plane, the orthographic interpolation will be used. Because the disparity becomes smaller with decreasing distance to the display plane, the maximum disparity for the orthographic interpolation is given by δ_O^d. Likewise, the distance d defines an upper bound for the disparity of δ_P^d, as the disparity decreases with an increasing distance to the display plane for DIBR with perspective elemental images, that are used for content with a greater distance to the display plane.

In addition, this solves the worst case scenario of the interpolation of orthographic images, as for content at infinity the perspective interpolation will be used that has a disparity of zero at that distance, as well as the worst case scenario of the interpolation of perspective images, as for content at the display plane the orthographic interpolation will be used that has a

disparity of zero at that distance.

The distance where the interpolation switches from orthographic to perspective images will be designed in reference to the perspective input images. This is because the evaluation of the interpolation of the two image representations, Chapters 4 and 5, revealed a higher overall fidelity for the interpolation of perspective images, due to the high angular resolution of the display and a relatively low spatial resolution. Another reason is the relatively low decline of image fidelity when the orthographic input images are reduced, see Chapter 4, which lead to the decision to decrease the orthographic images in favour of the perspective ones, making the perspective input images a determinant factor of the overall fidelity.

The distance where two diagonal neighbouring input images start to overlap is given by

$$d_{AOF} = \frac{\sqrt{2} \cdot \flat}{2 \cdot \tan(AOF)}, \tag{8.2.1}$$

depending on the diagonal distance between neighbouring elemental input images $\sqrt{2} \cdot \flat$ and the AOF of the elemental images.

The distance $d_{\mathcal{OP}}$ to switch between the different input images is doubled, such that every part of the interpolated frustum is sampled by at least two input views, when occlusions are not taken into account. This reduces the probability of artefacts due to occlusions, because a missing geometry can only occur, if a part of the interpolated frustum with geometry is occluded in at least two views. Finally, the distance $d_{\mathcal{OP}}$ to switch between the different input images is defined as

$$d_{\mathcal{OP}} = 2 \cdot d_{AOF} = \frac{\sqrt{2} \cdot \flat}{\tan(\psi/2)}. \tag{8.2.2}$$

The resolution of the orthographic and perspective elemental images are not considered because the number of orthographic views is directly correlated with the resolution of the elemental images (see Section 2.2.1 and Section 2.4.7). Another reason is that the footprint of one pixel depends on the physical lens system of the display and not the projection model (see Section 2.4.7). Finally, the spatial resolution of the display, see Section 2.2.1, is a constant for the device and equals the baseline $\flat$ between two neighbouring perspective elemental images.

8.3 Input Data

The input data consists of a subset of all orthographic and perspective colour and depth images, according to Chapters 4 and 5. The orthographic viewpoints are selected as an even distribution of the domain of all viewing directions, as described in Appendix C, and the perspective viewpoints are selected evenly distributed from the set of all elemental images, as described in Section 3.3.2. The ratio between orthographic and perspective input images is based on the results of Section 8.1, which are used to derive a minimum number of orthographic input images. The remaining available resources of the hardware are used to process the maximum amount of perspective input images.

8.4 Overview

The input is processed in accordance to the image representation. For every processed pixel the possibility of an early abort is checked, depending on the depth value. Afterwards, the interpolation for orthographic, and perspective images respectively, is applied, as described in the Chapters 4 and 5. The

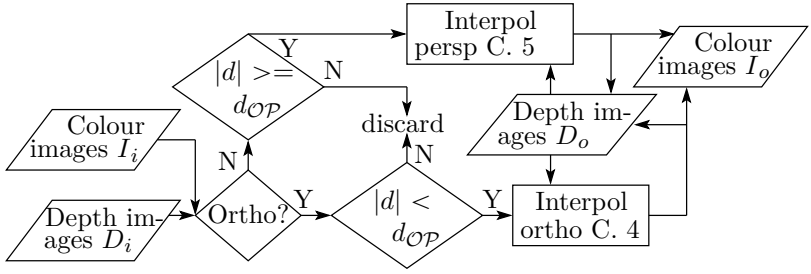

Figure 8.9. Overview of the interpolation algorithm for orthographic and perspective images.

orthographic interpolation already delivers perspective elemental images, therefore, the further processing is the same for all input images. The visibility problem is solved at the end of the interpolation by the z-buffer method, as described in Section 2.8.4.

8.5 Algorithm

In comparison to a static multi-view display, the resolution of a multi-view video display is relatively small. This allows to store a full frame buffer on the device. Under the constraint of a complete frame buffer, the processed order of the input images is arbitrary, as assembly of the interpolated output is done via depth tests. In addition, each interpolation method implements a depth test in a consistent coordinate system, allowing to use the colour output buffer as an accumulation buffer for blending of the interpolated output images, accumulating the weights, as well.

For parallel processing, the interpolation of one pixel has to have a blocking access to the colour, depth, and weighting buffers of the output pixel to ensure consistency of the depth values used for the depth tests and correct accumulation of the colour and weighting values.

8.5.1 Combining the Interpolation Results

In Section 2.4.7 the process of transferring the orthographic images into the elemental perspective images is described. The problem is tedious, because a single orthographic image is connected to all perspective images and vice versa. In case a full frame buffer is available, this problem is avoided, as the interpolated orthographic pixels can be written directly at the proper location in memory.

8.5.2 Adaptations for Parallel Execution

The interpolation algorithm is designed to be ported to a FPGA in the future and should allow for a modular composition of independent display devices. A fixed input data access pattern is achieved by limiting the interpolation to a maximum of the four nearest neighbours. The algorithm allows for a modular composition of display devices, because the input data depends only on the actively used part of the display plane. The warping algorithm allows for an efficient computation on specialised hardware, especially when disparity is used instead of depth, and instead of the L_2 norm, e.g. the L_1 norm is used for calculation of the costs. A further adaptation could be the use of a fixed-point number representation. Depending on the computational capability of the FPGA, the costly distribution of the colour information

could be omitted in the perspective case, which would result in a simplified blending weight.

One requirement for the efficient computation is a full frame and depth buffer. Hence, either the fraction of the display that one FPGA interpolates, or the resolution of the elemental images has to be adjusted to the available memory. The frame buffer is used for the accumulation of the colour values and transferring the orthographic representation into the perspective elemental image (cf. Section 2.4.7), because every elemental image depends on all orthographic images.

8.6 OpenGL Rendering

One reason perspective and orthographic images were selected as input for the interpolation algorithm is that the input data can be rendered with virtually every modelling tool.

Rendering of the input images for the interpolation with OpenGL has been described in Section 2.6.3 and Appendix A, in more detail. The perspective elemental images can be rendered according to the method proposed by Halle and Kropp [HK97], optional with a simulated fisheye lens, see Appendix A.3. Rendering of the orthographic input images is straight forward in OpenGL, but violates the assumption made in Section 2.4.7, regarding the perspective properties of single light rays. For large distances, the resulting oversampling is without consequence, since perspective images are used. For small distances, the result is a sampling with a lower frequency. The result of the proposed orthographic rendering is a sampling with a reduced frequency, resulting in a smoothed content very close to the display plane. This approach is justified by the small amount of viewing directions that are available for the orthographic interpolation, that could otherwise lead to interpolation artefacts caused by subsampling.

The proposed approach solves three challenges of rendering elemental images with OpenGL.

8.6.1 Image Distortions

The first is the viewpoint shift along the Z-axis by the distance of the NCP (see Figure 8.10) that leads to small image distortions, according to Halle and Kropp [HK97]. For the camera with the shifted viewpoint C', the

viewing rays that sample the two closest vertices are drawn as stippled lines, whereas the viewing rays of the camera at the original position C are drawn as dashed lines. The result of the viewpoint shift in Figure 8.10 is a slightly

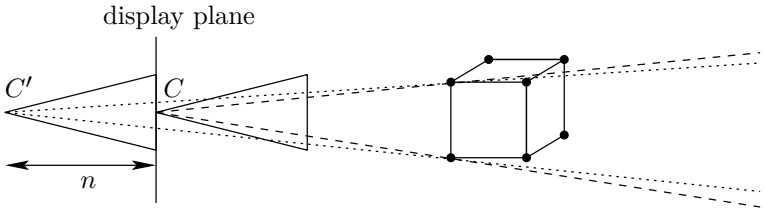

Figure 8.10. Distortion near the display plane. The effect is negligible for small viewpoint shifts, according to Halle and Kropp.

reduced distance in the image between the two sampled vertices, and rays, which sample the vertices under a different angle.

8.6.2 Z-Buffer Quantisation

In order to minimize the viewpoint shift, the NCP is placed very near to the camera centre, resulting in a coarse z-buffer quantisation and possible rendering artefacts like depth fighting, due to the non-linear depth sampling of the z-buffer, caused by the perspective division. The solution to this second challenge, proposed by Halle and Kropp, is to break up the frusta into multiple sections and render those in separate rendering passes. However, this is a tedious procedure that depends on the scale of the scene, and the amount of the viewpoint shift.

8.6.3 Geometry Artefacts

Holzbach and Chen [HC02] brought to attention that by shifting the viewpoint, another artefact might be introduced. For objects close to the camera centre, the viewpoint shift might cause parts of the objects to be rendered at the wrong position, or erroneously, as depicted in Figure 8.11. Object O is rendered in the orthoscopic and pseudoscopic image, although it is not located inside the viewing frustum of the elemental image. In the orthoscopic image, a part of the object is relocated to O', shifted by the uncorrected distance n of the orthoscopic camera C', occluding the view of

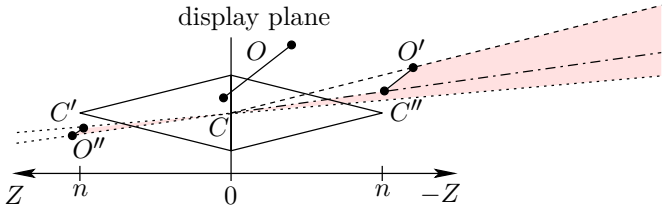

Figure 8.11. Clipping problem, solved by Holzbach and Chen using a scene analysis.

the background of the scene, as sketched by the filled area. The pseudoscopic camera C'' also samples the object, which is located in front of its NCP. Without a corrected depth, the result is a point reflected artefact at position O'', which occludes the filled drawn space. Holzbach and Chen solved this challenge with a scene analysis, shifting the camera centre slightly to avoid the rendering of such artefacts.

The proposed combination of orthographic and perspective images avoids the drawbacks associated with OpenGL rendering completely. The orthoscopic and pseudoscopic parts of the elemental images are rendered as described by Halle and Kropp [HK97]. The content near the display plane is rendered by the orthographic images, hence, the NCP does not need to be located at the display plane, allowing to omit the viewpoint shift. This solves the problems associated with the shifted camera centre, avoiding image distortions and rendering artefacts that originate from objects close to the display plane. Also, the NCP can be placed relatively far away from the display plane, allowing for a better depth sampling and less z-fighting.

8.7 Evaluation

The proposed algorithm, which combines the interpolated orthographic and perspective images, has been evaluated on both data sets. The sparse orthographic input images were selected from the evaluation in Section 8.1, whereas the perspective images were selected from the evaluation in Section 5.4. First, the fidelity of the interpolation is evaluated. Afterwards, the combined approach is compared against the interpolation solely using perspective input images (Section 5.4) and orthographic images (Section 4.4),

respectively.

In order to obtain comparable results, the same number of input rays is used for each of the three interpolation methods.

8.7.1 Coffee Capsules Data Set

For the combined approach, no ground truth background image has been used, in contrast to the interpolation of orthographic images. Especially, the interpolated orthographic input images that are used for the combined approach does not need a ground truth background image or a high fidelity in image regions of the background, as these are interpolated by the perspective input images.

The amount of input images that can be processed in real-time is limited by the resources of the hardware. Therefore, the full data set was interpolated from a total of 0.22% of all rays of the light field, consisting of 0.0367% orthographic input images and 0.1865% perspective input images. The ratio between the perspective and orthographic input images is based on the results of Section 8.1. The PSNR and the AD of all evaluated viewpoints are shown in the bottom row of Figure 8.12. The top row shows the viewpoint with the minimum PSNR and the centre row the viewpoint with the maximum PSNR, with the ground truth view, the interpolated view, and the negated difference from left to right. In the negated differences of view C_{103} and view C_{190} are some capsules with larger errors at the capsules border, which originate from the orthographic input images and indicate that these capsules are close to the display plane. When the interpolated view with the minimum PSNR of 27.7 is compared to the view with the maximum PSNR of 31.0, the perceived fidelity of the interpolated views is similar. In particular, no prominent artefacts are visible that interfere with the perception of the scene.

Next, the combined approach is compared against the interpolation of orthographic and perspective images, respectively. Each method interpolated the full data set out of 0.22% of all input views, whereas the combined approach used 0.0367% orthographic and 0.1865% perspective input images.

Figure 8.13 shows the PSNR in relation to the evaluated viewpoints and the mean PSNR of all viewpoints. The combined approach is plotted in green, the interpolation of perspective images in red, and the interpolation of orthographic images in blue. The combined approach has a marginal higher mean PSNR compared to the interpolated perspective images, while

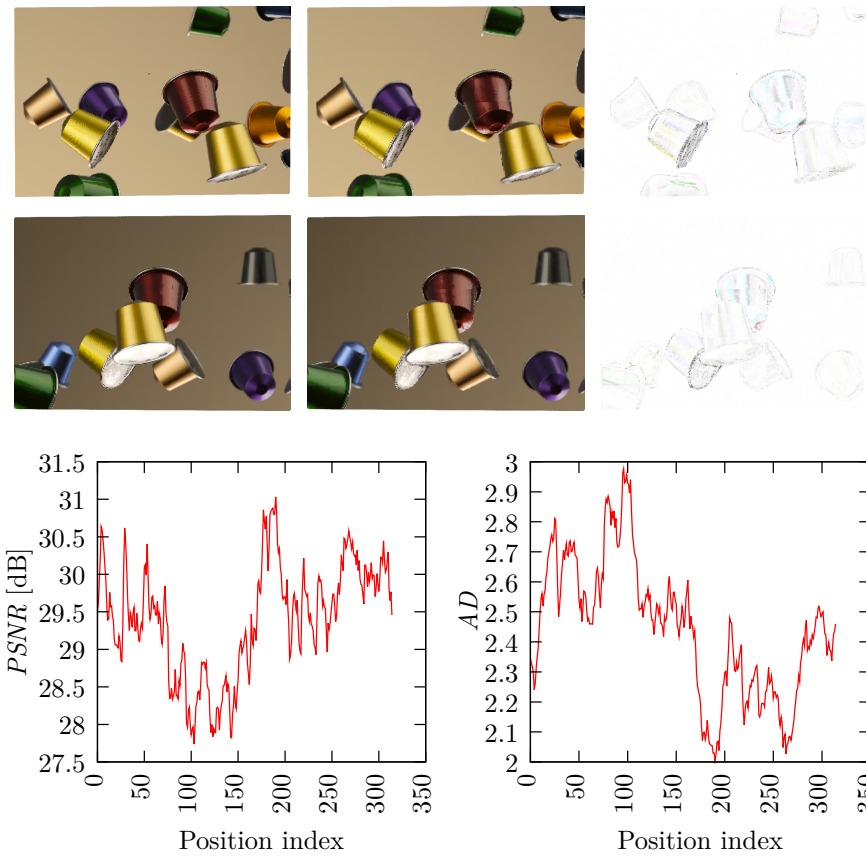

Figure 8.12. Upper row: Simulated view of position C_{103} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{190} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: *PSNR* (left) and *AD* (right) of scene *Coffee Capsules* with 0.22 percent of all rays as input. The axis of abscissae shows the viewpoint (cf. Figure 3.4).

the mean PSNR of the interpolated orthographic images is considerably lower. Table 8.1 shows details of the minimum, maximum, and mean PSNR of the combined approach $\mathcal{I}_c$, the interpolated perspective images $\mathcal{I}_p$, and

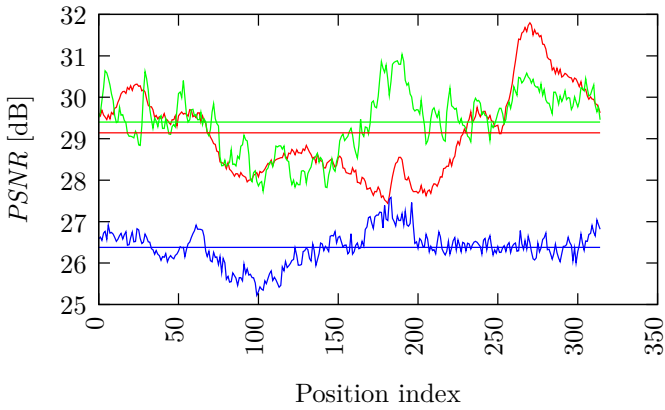

Figure 8.13. Line graph and mean *PSNR* of combined approach (green), interpolation of perspective images (red), and interpolation of orthographic images (blue) of scene *Coffee Capsules*.

the interpolated orthographic images $\mathcal{I}_o$.

Table 8.1. Results of *Coffee Capsules* scene showing the number of input pixels, the mean, the minimum, and the maximum *PSNR* of the sequence of viewpoints, as well as the position index C of the minimum and maximum *PSNR*.

Method	Input pixels [%]	Factor	Mean [dB]	Min. [dB]	Pos. C	Max. [dB]	Pos. C
$\mathcal{I}_p$	0.22	447.4	29.14	27.44	181	31.79	270
$\mathcal{I}_o$	0.22	447.4	26.38	25.22	99	27.58	183
$\mathcal{I}_c$	0.22	447.4	29.40	27.74	103	31.03	190

Near Position C_{181}, the viewpoint interpolated from the perspective images has the minimum PSNR and almost equals the PSNR of the interpolated orthographic images. The interpolated views and the negated difference to ground truth are shown in Figure 8.14. The simulated view C_{181} of the ground truth input images can be found in Figure E.1 of Appendix E. The combined approach has far less orthographic input images

Figure 8.14. Simulated view of position C_{181}, the minimum *PSNR* of the interpolation of perspective images.
Upper row: Perspective, combined, and orthographic interpolation (left to right).
Lower row: Negated difference to ground truth.

then the interpolation solely on orthographic input images, and also less perspective images than the interpolation solely on perspective images. This translates to a lower sampling near the display plane, when compared to the interpolation of orthographic images, and a lower sampling far away from the display plane, when compared to the interpolation of perspective images. Still, the combined approach outperforms the other methods from that view direction, due to a better overall sampling of the scene.

The interpolation of the orthographic images has a low overall PSNR on this data set, compared to the other methods, although this does not translate to a low perceived image fidelity. This is due to the many distinctive objects, which add up to a large amount of errors at the object boundaries, due to the low spatial resolution of the orthographic views.

In contrast, the perspective images have a relatively low error at the object boundaries, due to the high angular resolution of the elemental images. The areas with prominent interpolation artefacts disturb the perception of the scene, but does not affect the PSNR of the whole image, as much.

At Position 270 the interpolation of perspective images clearly outperforms the combined approach. From that position, the coffee capsule near

the display plane is occluded by a foreground capsule. The full set of interpolated views are shown in Figure E.3 of the Appendix E, as well as the viewpoint with the minimum PSNR of the interpolated orthographic images (see Section E.1).

8.7.2 Tutankhamun Data Set

The full set of images of the *Tutankhamun* scene were interpolated from a total of 0.28% of the data set, consisting of 0.2344% perspective input images and 0.0434% orthographic input images. Figure 8.15 shows the interpolated views with the minimum PSNR in the first row and the maximum PSNR in the second row. For each, the ground truth is on the left, the interpolated view is centred, and the negated difference to ground truth is on the right. The last row shows the PSNR for all evaluated views on the left, and the AD on the right side.

The viewpoint with the minimum PSNR shows relatively large errors at the floor, which reflects the window frame. This is a nontrivial interpolation artefact, which increases with an increasing baseline to the input view. The image information is transferred with the disparity that corresponds to the distance of the camera centre to the floor, which is correct for the amount of light that originates from the ambient and diffuse lighting of the floor. The reflected light, however, would have to be transferred with the disparity of the ray length that corresponds to the length of the reflected ray. Hence, the different reflected window frames from each of the input images does not converge to one, but to different reflections, due to a disparity that corresponds erroneously to the shorter distance to the floor, and is therefore too large. Except for the reflection on the floor, the interpolated images show no prominent artefacts, either on the foreground object nor on the background pillars or the window frame.

Figure 8.16 compares the combined approach against the interpolation of solely perspective and orthographic images, using the same amount of input images. The interpolation of orthographic images (blue) has about the same mean PSNR compared with the interpolation of perspective images (red). Nevertheless, for several interpolated viewpoints, the interpolated results show a large difference in image fidelity, up to eight dB.

The mean PSNR of the combined approach is about 2.5 dB higher than the interpolated images from a single image representation and outperforms the other interpolations for almost every evaluated viewpoint by a fair

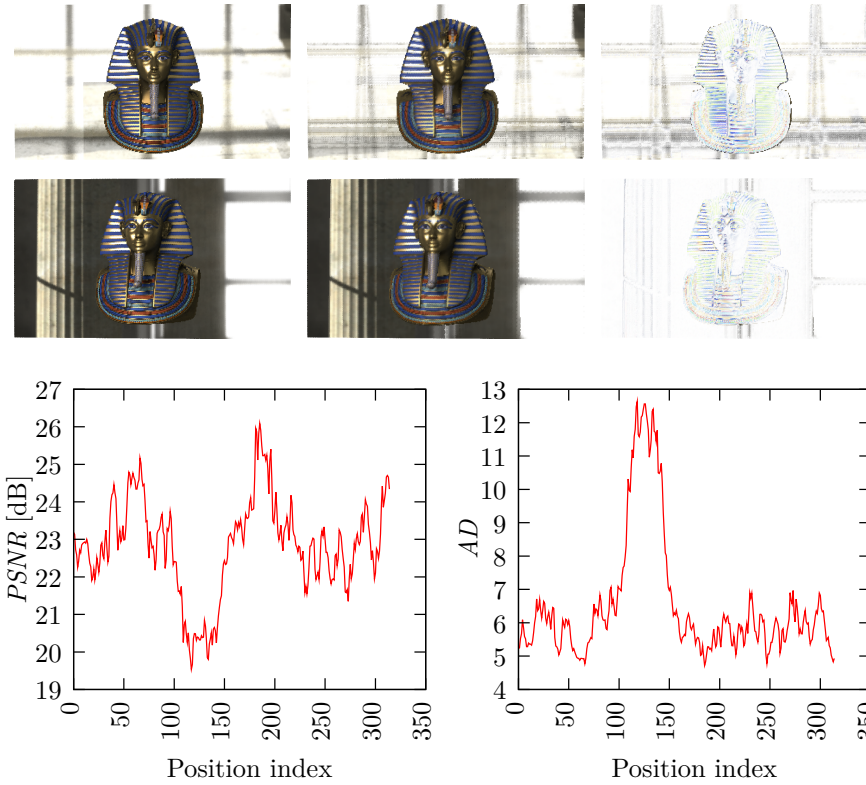

Figure 8.15. Upper row: Simulated view of position C_{117} (minimum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Middle row: Simulated view of position C_{185} (maximum *PSNR*, left ground truth, centre interpolated, right negated difference to ground truth).
Lower row: *PSNR* (left) and *AD* (right) of scene *Tutankhamun* with 0.22 percent of all rays as input. The axis of abscissae shows the viewpoint (cf. Figure 3.4).

amount.

The mean, minimum, and maximum PSNR of the interpolated views are summarised in Table 8.2, as well as the corresponding viewpoint index. At position C_{259}, the interpolation of orthographic images ($\mathcal{I}_o$) outperforms the other interpolation methods. That position corresponds to the maximum

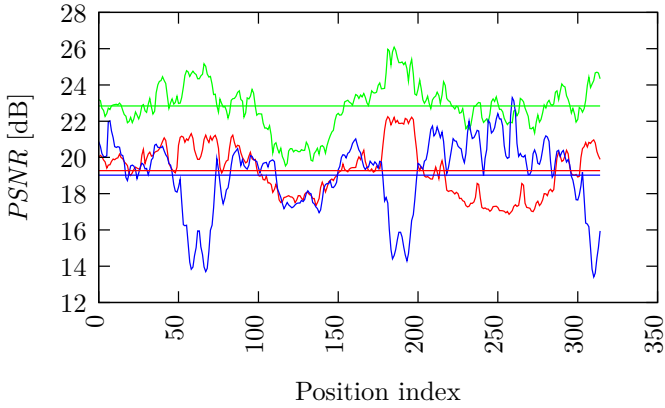

Figure 8.16. Line graph and mean *PSNR* of combined approach (green), interpolation of perspective images (red), and interpolation of orthographic images (blue) of scene *Tutankhamun*.

Table 8.2. Results of *Tutankhamun* scene showing the number of input pixels, the mean, the minimum, and the maximum *PSNR* of the sequence of viewpoints, as well as the position index C of the minimum and maximum *PSNR*.

Method	Input pixels [%]	Factor	Mean [dB]	Min. [dB]	Pos. C	Max. [dB]	Pos. C
$\mathcal{I}_p$	0.28	352.5	19.27	16.86	257	22.25	181
$\mathcal{I}_o$	0.28	352.5	19.02	13.40	310	23.29	259
$\mathcal{I}_c$	0.28	352.5	22.84	19.54	117	26.08	185

PSNR of the interpolation of orthographic images. The interpolated images for all three interpolation methods are depicted in Figure 8.17. The simulated view C_{259} of the ground truth images can be found in Figure E.4 of the Appendix E. When comparing the negated differences of the combined approach with the interpolated orthographic images, it can be seen that the background window frame shows distinctive artefacts in the interpolated orthographic images. The errors on the small image regions with structured

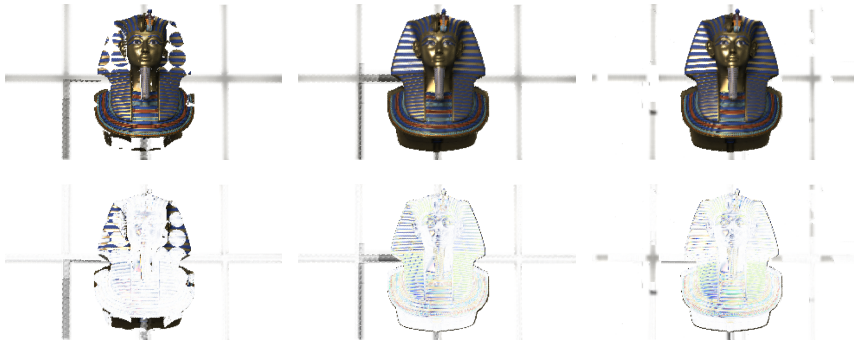

Figure 8.17. Simulated view of position C_{259}, the maximum *PSNR* of the interpolation of orthographic images.
Upper row: Perspective, combined, and orthographic interpolation (left to right).
Lower row: Negated difference to ground truth.

background are compensated by the higher fidelity of the interpolated foreground object, leading to a higher overall PSNR. However, considering that the combined approach uses just 0.0434% of the input images to interpolate the foreground object, in contrast to 0.28% of the input images that are used by $\mathcal{I}_o$, the gained fidelity is relatively small. Moreover, the perception of the scene is disturbed by the prominent artefacts of the background window frame of $\mathcal{I}_o$, which are not present in the combined approach $\mathcal{I}_c$.

The full set of interpolated views are shown in the Appendix E at Section E.2.

8.7.3 Runtime

The average time for the interpolation of one pixel on an Intel(R) Core(TM) i7 950 with 3.07 GHz is shown in Table 8.3. The runtime measurement excludes the load and write operations on the image data. The proposed interpolation I_c of the full *Coffee Capsules* data set would therefore require about 58 minutes and about 54 minutes on the *Tutankhamun* data set respectively, when time measurement is restricted to the interpolation. This translates to an acceleration factor of 1,527 for the proposed algorithm $\mathcal{I}_p$, when the interpolation of the colour images is compared to the rendering

Table 8.3. Average interpolation time for the interpolation of one pixel, compared to the rendering time of the ray tracer (RT). Below, the acceleration factors for the rendering of colour images are listed that do not take the ray traced depth and colour images into account. The acceleration factors for both evaluated scenes show the effective acceleration factors, which take the rendering of the input depth (RT_D) and colour (RT_C) images into account.

	Interpolation			**Ray Tracer**		
	$\mathcal{I}_p$	$\mathcal{I}_o$	$\mathcal{I}_c$	RT_C	RT_D	$\sum$
Avg. time [s/px]$\cdot 10^{-7}$	1.61	1.14	2.75	4,200	50	4,250
Acceleration factor				**RT input images**		
Rendering	2,609	3,684	1,527	[%]	[s/px]$\cdot 10^{-7}$	
Coffee Capsules	383	400	347	0.22	9.35	
Tutankhamun	310	322	287	0.28	11.90	

time of the ray tracer (cf. Table 8.3, lower left).

The data sets were interpolated with 200,000 views, which are roughly three orders of magnitude more views than current autostereoscopic displays offer (cf. Section 2.7.2). This translates to a rendering time of 3 to 4 seconds per frame, for a multi-view display with about 200 views. In order to achieve 21 FPS the rendering has to be accelerated by a factor of 63 to 84. The multi-threaded CPU reference implementation uses the computational more expensive depth for image warping, which will be replaced by computational less expensive disparity input images. This suggests that a careful implemented port of the algorithm to a FPGA could achieve real-time.

The percent of input pixels that were used for the different scenes are shown in the lower right of Table 8.3. For the proposed interpolation $\mathcal{I}_c$, in each case 0.04 percent of the input pixels were from orthographic images and the remaining from perspective ones. For the *Coffee Capsules* scene, the full colour data of $1.27 \cdot 10^{10}$ pixels of the display was reduced by a factor of about 440. The actual input data of the display was reduced by a factor of 330, when accounting for the input depth maps required by the interpolation algorithm. The rendering of the *Coffee Capsules* scene was accelerated by a factor of 347, if the rendering time of the input colour and

depth images are taken into account.

For the *Tutankhamun* scene, the full colour input data of $1.17 \cdot 10^{10}$ pixels was reduced by a factor of 350, and a factor of 260 respectively, when accounting for the depth images. Taking into account the ray tracing of the input colour and depth images, the proposed interpolation $\mathcal{I}_c$ accelerated the rendering of the full set of colour images by a factor of 287.

8.8 Conclusion

The presented approach of combining the interpolation results of perspective and orthographic images benefits of each image representation. It yielded good overall performance without artefacts that disturb the perception of the scene, neither near the display plane, nor at the background. Although this property can not be guaranteed for arbitrary scenes without a scene analysis, the proposed approach exhibits a good sampling of the scene, when compared to other generic input sampling methods. By applying the idea to image rendering with OpenGL, the beneficial sampling properties could be confirmed, since some of the drawbacks of previous approaches could be avoided.

The approach is limited to the maximum of the four nearest input images, hence the memory access is limited to a small amount of the input data and allows for spatial partitioned distributed computation of the interpolation, a prerequisite for scalable video displays.

Another prerequisite, derived from the real-time requirement, is that the computational costs for the interpolation should vary as little as possible, and be independent of the content, as well. In the proposed approach this is implemented by a relatively fixed interpolation scheme that copes without a costly scene analysis.

Along with the inexpensive interpolation algorithm, these properties should allow for a parallel execution on hardware that is tailored for the efficient processing of large amount of data, like a FPGA. Preliminary work of porting parts of the interpolation scheme to a FPGA gave reason to anticipate the feasibility.

The amount of input data for the display is reduced by a factor of 260 to 330, depending of the dimension of the simulated displays for the evaluated scenes, taking already into account the additional depth images, which are required for the interpolation. Given the hypothetical full parallax

multi-view display of Section 2.7.2 with 40 x 40 views in HDTV, three colour channels, and 21 FPS, the achieved compression factors allow to reduce the uncompressed 195 Gbytes per second to the range of 4.7 Gbit/s to 6 Gbit/s. This data range is well within the specified data rate for dual link Digital Visual Interface (DVI), which is up to 7.44 Gbit/s, therefore allowing to operate the hypothetical HDTV multi-view display with a single work station.

For transfer over a network, these data rates are still a challenge. Assuming a mean data reduction factor of 295, the hypothetical display will have a data rate of 39.7 Gbytes per minute, which accumulates to 2.32 Tbytes in an hour. Taking lossless compression into account, for typical images an additional compression factor of 10 can be assumed, resulting in amounts of data that are still cumbersome to transfer over a network. However, for content that can be rendered with OpenGL it is sufficient to transfer a scene description over network, and render the input images on the computer that is linked to the display, as described in Section 2.6.3. Scene descriptions are usually a lot smaller then the rendered videos, which should apply even more for multi-view displays, and do not depend on the number of images that are rendered, but on the content of the scene. Therefore, the proposed rendering from an OpenGL scene description is an appealing solution to transfer the content for multi-view displays over a network, but has a very limit application.

Chapter 9

Conclusion

In the last few years, technological advances allowed for full parallax displays to emerge on the market, based on the principles of integral imaging, which were discovered in the early 20^{th} century by Gabriel Lippmann. The increasing number of horizontal and vertical views are accompanied by a vast amount of data that has to be rendered, transferred and stored. In case of full parallax video displays, an additional challenge is the real-time data transmission to the display device, which quickly exceeds the capabilities of off-the-shelf consumer hardware.

9.1 Summary

In this work, DIBR was utilised to interpolate the full data set out of few input images, pointing out an approach to reduce the vast amount of input images to a few samples. This reduced the number of colour images, and thereby the amount of data, but introduced a dependency on precise depth images in order to yield high quality interpolation results.

Starting with a basic DIBR algorithm, the properties of the interpolation and the effects of reducing the number of input images were investigated for both, orthographic and perspective image representations of the light field. It has been shown that the location and number of sampling points are a key factor to image interpolation, when noticeable artefacts are to be avoided, and it was also shown that both depend on the geometry of the scene.

For full parallax poster displays, a DIBR algorithm was introduced that has been tailored to exploit the parallel capabilities of GPUs and multiprocessor CPUs of modern workstations. The challenge of input image selection was met by a best-next-view selection algorithm that produced an optimised ranking of the input image locations, taking into account

the sampling of the scene and the angle, under which points in the scene were observed. The rendering of the elemental images were accelerated by a factor of 400 by the implementation on the GPU, when compared to rendering with a ray tracer. Taking into account the computation time of the BNV selection and the ray tracing of the input images, an effective acceleration factor of about 100 remained. This approach allowed to render the complete data set within days on a single workstation, in contrast to months of rendering time when the complete data set is rendered on a ray tracer.

Section 2.7.2 introduced the required data rates for autostereoscopic multi-view video displays. The presented approach to solve the challenge of data transmission was to use DIBR algorithms and interpolate the image data on the device, in real-time. Therefore, the number of input images had to be reduced even further, and the interpolation algorithm had to be computational inexpensive with limited memory access patterns. These constraints eliminate the possibility of a scene analysis, although the interpolation algorithm should be able to handle arbitrary scenes without introducing prominent artefacts. The challenge was met by combining the interpolation results of different image representations, exploiting the depth dependent benefits offered by both image representations. This allowed to reduce the amount of input data to a fraction of about 0.30% to 0.38%, without introducing artefacts caused by occlusions or unseen areas of the scene, and avoiding a costly scene analysis.

The rendering of the colour images was accelerated by a factor of 1,527, when comparing the multi-threaded CPU implementation of the interpolation with multi-threaded ray tracing. The interpolation algorithm is computationally inexpensive and the CPU reference implementation can be further simplified, to allow an efficient implementation on a FPGA. This gives reason to expect that an implementation on a FPGA can accelerate the interpolate by an additional factor of 63 to 84, and therefore render the content in real-time. Although a full sampling of the scene is not guaranteed by the proposed approach, it could be shown that the approach performed well on challenging scenes of a typical application with large depth variety, multiple objects, and a prominent foreground object, as well.

The same principle was applied to avoid three pitfalls, when rendering elemental images for full parallax displays with OpenGL. The first two pitfalls are related to shifting the viewpoint and the NCP. According to Halle and Kropp [HK97], the image distortions for small viewpoint shifts

are barely noticeable, and can therefore be neglected. The artefacts that are caused by point reflected objects near the display plane, as described by Holzbach and Chen [HC02], interfere with the perception of the scene and are avoided by the proposed rendering without a scene analysis. The third pitfall is a coarse depth quantisation, due to a very close NCP, which is required to minimize the viewpoint shift.

9.2 Future Work

In this work, the consequences of a very sparse set of orthographic input images were investigated, under the constraints of a fixed number of perspective input images, whose baseline between neighbouring input views was used to derive the distance were the different interpolated results should be swapped. An evaluation of a very sparse set of perspective images, combined with a very sparse set of orthographic views, may allow to further reduce the number of input images to a fraction of the input images used for the evaluation in this work. Because this evaluation introduces a varying distance for swapping the interpolation results, this has not been evaluated.

Porting the proposed DIBR algorithm to a FPGA in order to evaluate the real-time performance is another task left for the future. First implementations show the feasibility of the pixel warping algorithm, and sufficient data rates of the DVI interface and the Random-Access Memory (RAM), available at the FPGA development platform.

A very small number of input views may allow for a BNV analysis of the input views in order to reduce potential occlusions of the background by foreground objects, which may occur due to the restriction to the closest four input views. However, at this early stage of an implementation on the FPGA it is not investigated, if dropping the current fixed memory access pattern is feasible under the real-time constraint.

Appendix A

Content Rendering with OpenGL

Halle and Kropp [HK97] introduced an algorithm for efficient rendering of images for full parallax displays that takes advantage of conventional graphics hardware and OpenGL.

A.1 Rendering of Elemental Images

In order to achieve this goal, the rendering process had to be adapted to fulfil the special requirements of image rendering for full parallax displays. In the following, the rendering algorithm of Halle and Kropp is introduced in detail.

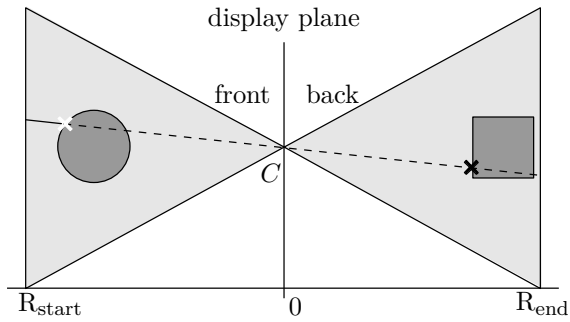

Figure A.1. Viewing frustum of an elemental image of a full parallax display (sketch in 2D).

The viewing frustum for one elemental image of a full parallax display is depicted in Figure A.1. The image for a lens system of the display is

rendered by placing a camera into its position. Then a ray is cast for every viewing ray of the elemental image, starting in front of the display at a defined distance to the virtual display plane. At the position of the lens system, the camera centre, the foreground geometry is point reflected and, therefore, appears upside down in the elemental image with left and right inverted. The ray ends at the specified maximum distance behind the virtual display plane. If it is assumed that a viewing ray does not change along its path, the intersection of the viewing ray with the model is cancelled at the first intersection point, marked by the white cross in Figure A.1.

In order to render content for the full parallax display with the OpenGL library, the algorithm introduced by Halle and Kropp divides the viewing frustum of the elemental image into two parts. The first part is behind the display plane and can be rendered directly with OpenGL (see Figure A.2, right).

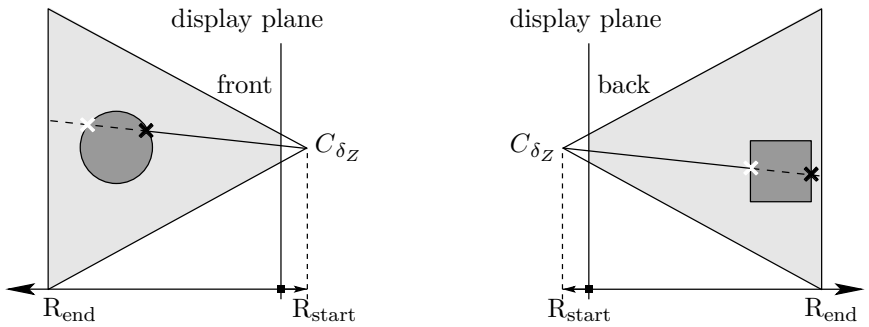

Figure A.2. Viewing frustum in front of (left) and behind (right) the virtual display plane.

In the OpenGL camera model, the near clipping plane can not be set to the centre of the camera. To avoid this limitation the camera centre is slightly shifted backwards along the optical axis, in order to place the near clipping plane on the virtual display plane. According to Halle and Kropp, this imposes the drawback of small image distortions and a potentially coarse depth resolution, due to the small distance of the near clipping plane to the camera centre. They concluded that these problems have very little impact on the final image and the coarse depth resolution could be avoided by separating the viewing frustum into smaller pieces.

The second part is the rendering of those parts of the scene, which are in front of the virtual display plane. Therefore the camera is rotated to face the opposite direction and shifted backwards along its optical axis, such that the near clipping plane lies at the virtual display plane. Figure A.2 (left) shows the frustum of one elemental image that lies in front of the display. In order to achieve the same rendering behaviour as in Figure A.1, the intersection test has to choose the intersection that has the greatest distance to the camera centre, marked by a white cross in Figure A.2 (left), as opposed to the closest intersection, marked by a black cross. In OpenGL this can be achieved by setting the function that compares the depth values to prefer greater values and by clearing the depth buffer to zero (see Listing A.1).

Listing A.1. *Adjustment of the depth order for rendering of pseudoscopic images with OpenGL.*

```
glDepthFunc(GL_GREATER);
glClearDepth(0.0);
```

The surfaces, that are nearest to the camera, lie on the back side of the objects. Hence, it is important to turn back face culling off or to set OpenGL to cull front faces of objects.

For the remainder of this section, the part of the scene that lies in front of the virtual display plane is called *front geometry*, and the part that lies behind the virtual display plane is called *back geometry*. The scene is divided by the virtual display plane into the front and back geometry, which are rendered separately. The front geometry will be rendered to produce a pseudoscopic image, whereas the back geometry will be rendered orthoscopic. In order to account for the point reflection, the pseudoscopic image coordinate axes have to be flipped, before overlaying it with the orthoscopic image.

Both images can be either combined by clearing the z-Buffer after rendering of the orthoscopic image and before rendering of the pseudoscopic image, leaving the colour buffer intact, and flipping the viewport axes to account for the point reflection, or by rendering both images to a texture and compute the final result in a third render pass. This is done by texturing the rendered images on a viewport aligned quad. By rendering with an orthographic projection it is ensured that the pixels of both textures map exactly on the viewport of the final image. Both images are combined by a fragment shader. In this case the changes to the viewport can be superseded

by correcting the texture coordinates of the pseudoscopic image, to account for the point reflection.

A.2 Reflection Model and Shading

Halle and Kropp did several adjustments to correct the lighting for the rendering of pseudoscopic images with the Phong reflection model. Figure A.3 shows an Object O, placed in front of the virtual display plane, where the camera centres C_1 to C_3 are located. For content in front of the display, the cameras render pseudoscopic images (solid drawn cameras), and for content behind the virtual display plane, orthoscopic images are rendered (stippled drawn cameras).

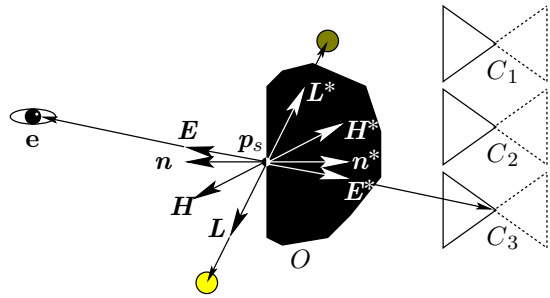

Figure A.3. Phong reflection for pseudoscopic image generation with *OpenGL*.

The Blinn-Phong reflection model, which is the standard in OpenGL, has been introduced by Phong [Pho75], Blinn and Newell [BN76]. Figure A.3 shows the Blinn-Phong reflection model for the surface point $\boldsymbol{p}_s$, viewed by the observer **e**. The reflection model uses the normalised direction vectors of the light source **L**, the direction to the observer **E**, the direction half between the direction to the light source and the direction to the observer **H**, and the surface normal **n**.

Due to the modified visibility test (see Listing A.1), $\boldsymbol{p}_s$ will be visible in the pseudoscopic image of C_3, which renders the light ray that is observed at position **e**. For correct lighting from position C_3, however, the normalised direction vectors in Figure A.3 have to be point reflected in $\boldsymbol{p}_s$, denoted by the superscripted star, yielding a mathematical equivalent result for $\boldsymbol{p}_s$.

Hence, the implementation of pseudoscopic rendering is straight forward for Gouraud shading [Gou71] using vertex shaders, the standard shading in OpenGL. Gouraud shading evaluates the reflection model only at the vertices, in between the results of the confining vertices are interpolated. Hence, for the rendering of, e.g. specular highlights, in particular when vertices are far apart, Gouraud shading may yield rendering artefacts, in this example vanishing and appearing highlights, when the observer moves in front of the display. This flickering can be avoided by a per pixel evaluation of the reflection model, using Phong shading. Phong shading can be implemented by computing the vectors for the reflection model in a vertex shader for every vertex, and use of the rasterizer to obtain interpolated shading of those vectors for every fragment. In the fragment shader, the reflection model is evaluated per fragment, yielding correct and consistent highlights for a moving observer.

A.3 Implementation of a Fisheye Projection in OpenGL

So far, rendering is limited by the camera models of OpenGL. Some displays may require a special lens model for content generation, which is needed to sample the artificial scene with the viewing rays of the display. In the following it is described how a vertex shader can be used to simulate an equidistant fisheye projection (see Section 2.4.6) that is used to project the different viewing rays of an elemental image of the full parallax display. The idea is to use a vertex shader to move the vertices to a position, where the image rendered with OpenGL equals an image taken with an equidistant fisheye camera. This modification has to be done in dependence of the viewpoint, because this lens effect is achieved by modifying the geometric model of the scene.

As long as the vertices of an object are reasonable closely grouped, this will approximate the fisheye projection well, since between the vertices bilinear interpolation is performed during rasterization. Larger deviations from the fisheye projection may occur when vertices of an object span over large areas of the image, such that the bilinear interpolation is not appropriate anymore. In this application the FOV of the camera used for rendering is 40 degree, hence, the deviation between a perspective projection

and the equidistant fisheye projection is small, and errors introduced by bilinear interpolation are not prominent. For cameras with larger FOV, a sparse geometry could be remedied by utilising a *tessellation* shader to create a higher level of geometric detail. The tessellation shader is invoked after the vertex shader, therefore the correction to the vertex position would have to be relocated to the geometry shader.

First, the model-view matrix is applied to yield camera coordinates for the current elemental image and the projection matrix is set to identity. Afterwards the vertex position (x, y, z) is transformed into spherical coordinates by

$$\varphi = \arctan\left(\frac{\sqrt{x^2+y^2}}{z}\right) \quad \text{and} \quad \theta = \arctan\frac{y}{x}\,. \tag{A.3.1}$$

The FOV (ψ) of the fisheye camera is accounted for and the new vertex position $(x^{'}, y^{'}, z^{'}, w^{'})^T$ is assigned by

$$\begin{pmatrix} x^{'} \\ y^{'} \\ z^{'} \\ w^{'} \end{pmatrix} = \begin{pmatrix} \frac{\varphi}{\psi} \cdot cos(\theta) \\ \frac{\varphi}{\psi} \cdot sin(\theta) \\ \frac{z+n}{f-n} \\ 1 \end{pmatrix}\,. \tag{A.3.2}$$

Appendix B

Simulating Multi-View Displays

B.1 Overview

An overview of the algorithm to render virtual views of a multi-view display is given in Figure B.1. The simulation consists of three steps, beginning with the configuration of the multi-view display that is used to build a virtual model. Then, a database is built that maps the viewing rays of all viewpoints to the image coordinates of the elemental images of the intersected lens systems. Finally, all elemental images are read and the colour is assembled in the output images, utilising the image coordinates that were obtained in the previous step.

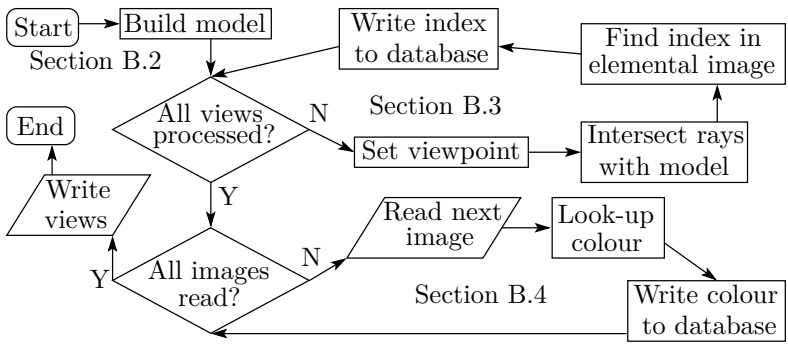

Figure B.1. Overview of the simulation algorithm for a multi-view display.

B.2 Simulation of the Display

The physical display is simulated by building a virtual model of the display, allowing to define the number of lens systems in width and height, and the distance between the lens systems d_{lp}, therefore allowing to define the spatial resolution of the display (see Section 2.2.1).

Each lens system is represented by a sphere, allowing for an efficient implementation of the intersection with the viewing rays (see Figure B.2). The size of the aperture of the lens system is simulated via the size of the spheres, a parameter that relates to the pixel size d_p of conventional 2D displays.

The angular resolution (see Section 2.2.1) is defined via the projection model of the lens system (see Section 2.4.6). Changing the angular resolution by the number of views requires a re-sampling of the light field via the elemental images, which is described in Section 3.3.3.

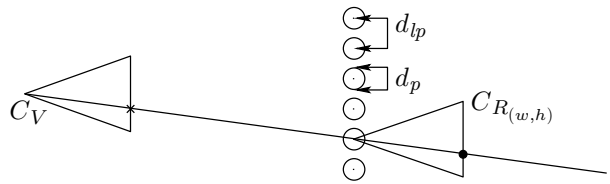

Figure B.2. Simulation of the physical display by placing a sphere for each pixel of the spatial resolution of the display.

B.3 Building a Ray Index for the Virtual Views

A moving observer is defined by a sequence of viewing transformations, allowing to define the position and orientation of the observer. For each viewpoint C_V, the camera is placed and a ray cast is made, intersecting the model of the display with the viewing rays. For each intersection, the position of the observer C_V is projected into the image plane of the elemental image of the intersected lens system $C_{R_{(w,h)}}$, denoted by a black dot in Figure B.2, yielding the image coordinates of the elemental image. The

projection model of the lens systems has been introduced in Section 2.3 and is now utilised to map the viewpoint to the view depending colour information. The image coordinates of the viewing ray (the black cross in Figure B.2) and the image coordinates of the elemental images, which encodes the colour of the viewing ray, are saved in the database.

A rendered view of the whole display depends on all elemental images of the display (cf. Section 2.4.7). Hence, building the database is necessary to avoid reading the light field multiple times from disk, as it does not fit into memory.

B.4 Rendering of the Views

After all viewpoints are processed, the simulated views are rendered. All elemental images are read, and for each image the corresponding views are looked up in the data base. The colour information is extracted from the elemental image using the stored image coordinates and bilinear interpolation, and assembled in the simulated views at the proper position. After all elemental images have been processed, rendering of the simulated views is completed and the result is written to disk.

Figure B.3. Simulated view of the display showing the *Tutankhamun* scene (left) and the *Coffee Capsules* scene (right). The observer is placed centred, 6 meters in front of the display.

The Figures B.3 show a simulated view of both data sets. The viewer is centred 6 meters in front of the display, the pixel pitch of the display is set to 2 mm and the size of each pixel is set to overlap with its neighbours, to avoid background pixels in the active display area. The cause of the

darker rendered background at the lower left side of the left figure is due to the mentioned re-rendering of a part of the data set, as described in Section 3.1.2.

For content close to the virtual display plane, aliasing can be observed at the colour ornamentation and at the borders of the coffee capsules in the simulated ground truth views of Figures B.3. The aliasing is due to a high frequency in the scene, sampled by the relatively low spatial resolution of the lens systems of the display. For content near the display plane this does not allow for a smooth transition between neighbouring lens systems (see Section 2.9.5). This kind of aliasing would also be visible on real displays and occur in all spatially sparse multi-view displays.

B.5 Notes on Efficiency and Rendering

For each simulated view, the complete view dependent colour information of each pixel of the display has to be processed, which usually results in the whole data set being processed. The computation time for the intersection tests and the projection of the viewpoints in order to extract the view dependent colour information is relatively small in comparison to the amount of time required to read all images of the data set from the hard disk drive, which takes approximately two hours. In order to render the path of an observer, many views have to be simulated, and processing of the whole data set for every view is not an option.

This problem is resolved by first building the data structure that maps the pixel of the simulated view to the pixel of the display and processing as many views as fit into the memory. Then, the whole data set is processed to retrieve the view dependent colour information and the simulated views are written to the hard disk drive. Afterwards, the next batch of views are processed, until all views are simulated. This approach allows to render short simulated sequences, while processing the data set only once.

Further acceleration was archived by parallelization using OpenMP, which was straight forward in this case. In the second step the intersection of all viewing rays of the simulated view was parallelized. To increase utilisation, a dynamic schedule has been used with a lock when changes on the data structure were made.

The simulation offers an easy way of incorporating effects from the lens system like, e.g. a circle of confusion, into the simulation. Furthermore

some parameters of the lens system can be changed via the projection model, e.g. the FOV or accounting for a radial distortion. Other effects can be applied by pre-filtering of the images of the data set, e.g. applying a Gaussian filter.

Appendix C

Subsampling of Orthographic Input Images

In order to evaluate the viewpoint interpolation with a different number of input views, it is described how the input images and the distribution of the input images have been acquired. In the orthographic representation, viewpoints are described by the angle of the light rays, passing through the display plane (see Section 2.4.7). An equidistant fisheye projection describes the mapping between a pixel in an elemental image and the angle of that viewing ray. Hence, in a perspective image, one pixel corresponds to an orthographic view of one viewing direction. The problem of distributing a number of viewpoints evenly can therefore be reformulated to the even distribution of points inside the active, circular area of a spherical 2D image. In order to formulate a general solution, a unit circle is placed in the coordinate centre and mapped to the resolution of the 2D image after the solution has been computed. Before a solution can be computed, the problem has to be formulated, according to the requirements of the solver. As solver, an algorithm introduced by Levenberg [Lev44] and Marquardt [Mar63] has been chosen, which computes the minimum of a function in a gradient descent manner. Solver using a gradient descent usually do not guarantee to find the global minimum, but can be trapped in a local minimum. For this reason it is required to start close enough to the global minimum.

C.1 Generating a Start Distribution

A start distribution is set by randomly inserting the viewpoints into the 2D image. Afterwards, two conditions are checked. The first condition is that all viewpoints lie inside the circular area and the second condition is that all points have a minimum distance to all their neighbours that is greater than

zero. Points that do not fulfil those conditions are removed and reinserted with new random values, until both conditions are fulfilled. In the next step, the viewpoints should be shifted to achieve a regular spaced distance pattern between neighbours. This problem has to be modelled in order to compute a solution.

C.2 Modelling the Problem

The problem is modelled by introducing a repelling force f_r between the viewpoints, similar to the repelling force between two electrons, which degrades with an increasing distance d_f between two points

$$f_r(d_f) = \frac{1}{2 \cdot {d_f}^2}. \tag{C.2.1}$$

Equation C.2.1 is plotted on the left side of Figure C.1 and has a singularity by $d_f = 0$. Due to the constraints put on the start distribution it is assured

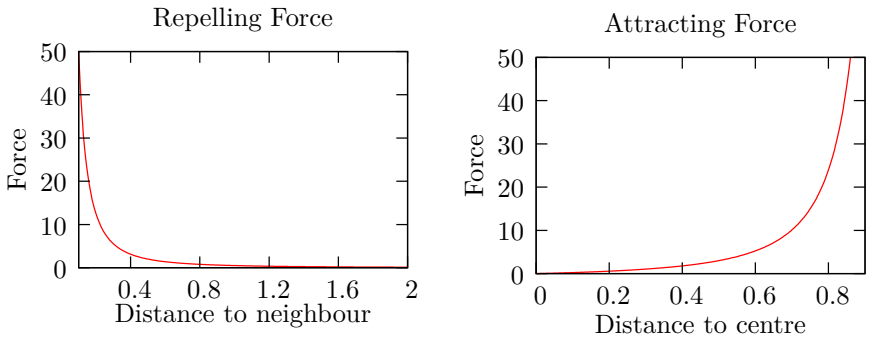

Figure C.1. Force functions used to define an inward pushing force (right side) and a repulsion against other particles (left side).

that all viewpoints have a minimum distance to their neighbours that is greater than zero.

Without a counteracting force, the viewpoints would simply drift apart, leaving the circular area of the solution. This is prevented by a force f_a that attracts the points to the centre of the circular area and increases with

the distance of a point to the centre of the circle $\mathbf{c}$

$$f_a(\boldsymbol{c}) = \frac{1}{(1-\boldsymbol{c})^2} - 1. \tag{C.2.2}$$

The force is plotted on the right side of Figure C.1 and has a singularity by $\boldsymbol{c} = 1$. The first constraint of the start distribution assures that all points have a distance smaller than one to the centre of the circular area.

Appendix D

Efficient Ray Cast Implementation

The ray cast has been implemented using different approaches. The reference implementation is implemented single threaded on the CPU, utilising only one of the available kernels. Listing D.1 shows the pseudo code for rendering of an image via ray cast. Every viewing ray of the elemental image is cast and its intersection with the model is calculated. The first LFN that is intersected by the viewing ray is then used to, e.g. interpolate the colour from the angular closest colour samples of that LFN.

Listing D.1. *Pseudo code for the ray cast of an image*

```
for (every row y) {
  for (every column x) {
    ray = MakeRay (y, x)
    LFN = ocTree->Intersect (ray)
    outPixel[y][x] = LFN->InterpolateColour (ray)
  }
}
```

Ray casters are particularly suited for parallelization, as all viewing rays are processed independently, although there are implementations, which exploit image and object space coherence [WSBW01].

D.1 Multi-threaded Implementation on the CPU

The parallelization on the CPU has been implemented using OpenMP, which offers an easy way to utilise multi-core processors. For rendering of the

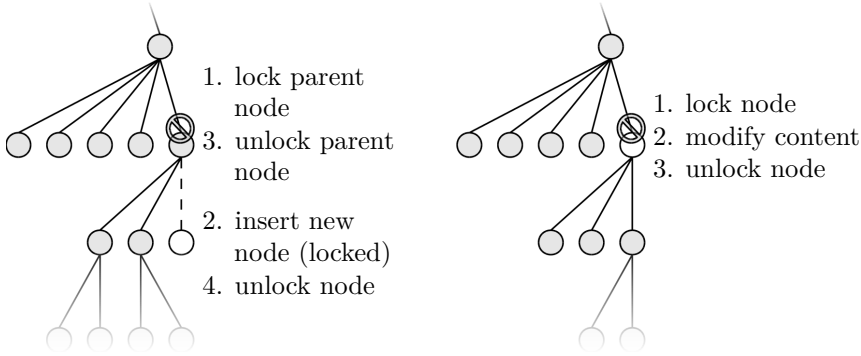

Figure D.1. Insertion (left) and modification (right) of an octree's node.

interpolated images, the parallelization is straight forward because access on the data structure is read only. Listing D.2 shows how OpenMP can be used to parallelize the outer rendering loop to take advantage of the multi-core processor.

Listing D.2. *Parallelization of the outer rendering loop with OpenMP*

```
#pragma omp parallel for schedule (dynamic)
for (every row y) {
  ...
}
```

The other parts of the algorithm that utilise the ray cast modify the data structures, and therefore, need a locking mechanism when executed in parallel. The operations, that need to be made thread safe, are the insertion of new points, and the modification of existing nodes. The locking mechanism is implemented by an OpenMP lock and each node of the octree is given a lock to guarantee exclusive access. An OpenMP lock can only be held by one thread at a time, and if more than one thread attempts to acquire the lock, all, but the thread that holds the lock are paused. When the lock is released by the thread that holds the lock, the next thread can acquire it.

The procedure of inserting a new node into the octree is depicted in Figure D.1 (left). The node that should be inserted is dyed white and connected with its parent node by a dashed line. New nodes are either

inserted as leafs to hold a LFN or as inner nodes for hierarchical space partitioning. To guarantee that the thread that inserts a node has exclusive access to the data structure, the parent node is locked. Then, the new node is created and locked by the thread, which holds the lock of the parent node. The new node is attached to its parent node and both locks, which are held by the thread, are released.

Figure D.1 (right) shows a situation where the node or its content should be modified by a thread. The node that should be modified is dyed white. Then, the node is locked, its content modified, and afterwards the lock is released by the thread. The data structure of the LFN is attached to the octree as content and is thread safe, because access to the content of the octree is guaranteed to be exclusive by its locking mechanism.

Accelerating the rendering with OpenMP on the CPU is straight forward and takes full advantage of the hierarchical data structures, but is limited by the number of available CPUs. In order to further accelerate the algorithm, an approach has been implemented to accelerate the intersection of the viewing rays of the display with the geometric model on the graphics hardware, based on the work of Weghorst et al. [WHG84].

D.2 Accelerating on the Graphics Hardware

In OpenGL, the parts of the scenes in front of the display plane and behind the display plane have to be processed separately, as described in Appendix A. Every LFN gets a unique identification number assigned, which is encoded as a 24-Bit colour. This mapping between the LFN and a unique colour will be used to render the geometric model, and identify the LFN that is hit by the viewing ray. The colour *black* is assigned to the background of the scene and is used to identify viewing rays, which have no intersection with the geometric model. Each LFN belongs either to the front geometry or the back geometry. Therefore, the mapping of the front geometry of the scene to a colour is independent of the back geometry's mapping, and colour values can be reused.

For rendering, each LFN is represented by a quad in the associated colour. The front and back geometry are rendered separately to different textures with the previously described modifications of the rendering. In a last rendering pass, both images are combined. This is done by texturing the rendered images on a viewport aligned quad. By rendering with an

orthographic projection, it is ensured that the pixels of both textures map exactly on the viewport of the final image. The pseudoscopic front geometry image is side-corrected via separate texture coordinates that are mapped onto the quad.

Both images are combined by a fragment shader. Those parts of the front geometry's image that have the black background colour are filled with the values of the back geometry's image. The result is a pseudo-coloured image of the scene where the intersections of the viewing rays with the geometric model are obtained and encoded in the mapping of the colour to the LFNs. The final image is generated on the CPU in parallel by identifying the intersected LFNs and by interpolating the colour value from the angular closest colour information, that is saved by the LFN.

Appendix E

Additional Results

E.1 Coffee Capsules Data Set

Figure E.1. Simulated ground truth view of positions C_{99} (left), C_{181} (centre), and C_{270} (right).

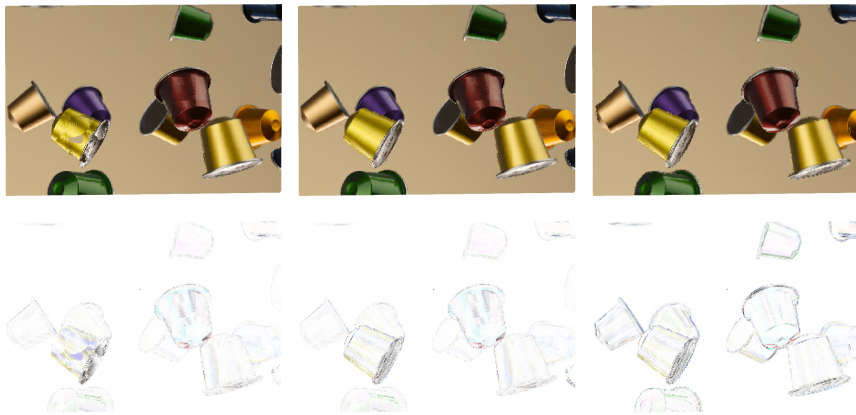

Figure E.2. Simulated view of position C_{99}, the minimum *PSNR* of the interpolation of orthographic images.
Upper row: Perspective, combined, and orthographic interpolation (left to right).
Lower row: Negated difference to ground truth.

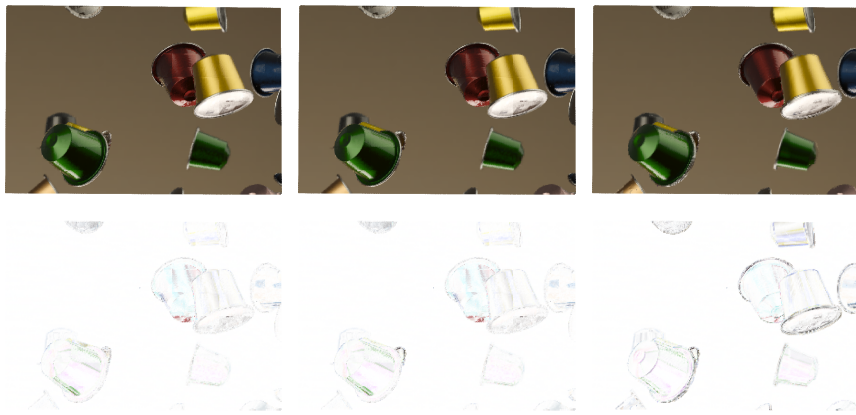

Figure E.3. Simulated view of position C_{270}, the maximum *PSNR* of the interpolation of perspective images.
Upper row: Perspective, combined, and orthographic interpolation (left to right).
Lower row: Negated difference to ground truth.

E.2 Tutankhamun Data Set

Figure E.4. Simulated ground truth view of positions C_{181} (left), C_{259} (centre), and C_{310} (right).

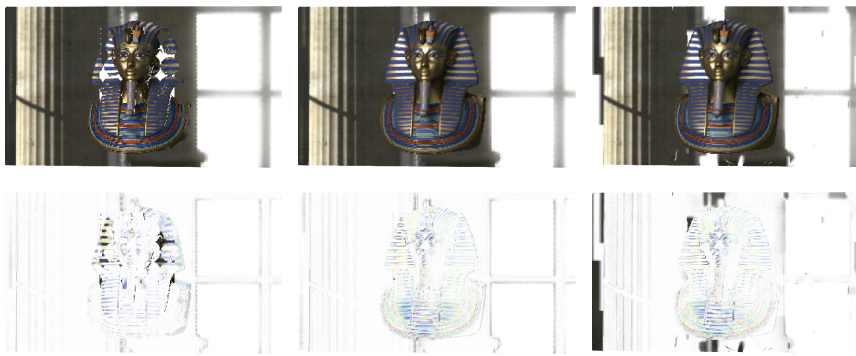

Figure E.5. Simulated view of position C_{181}, the maximum *PSNR* of the interpolation of perspective images.
Upper row: Perspective, combined, and orthographic interpolation (left to right).
Lower row: Negated difference to ground truth.

Figure E.6. Simulated view of position C_{310}, the minimum *PSNR* of the interpolation of orthographic images.
Upper row: Perspective, combined, and orthographic interpolation (left to right).
Lower row: Negated difference to ground truth.

Bibliography

[AB91] Edward H. Adelson and James R. Bergen. The Plenoptic Function and the Elements of Early Vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, 1991.

[ABCO+01] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *Proceedings of the conference on Visualization*, VIS, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[Abr03] Albert Abramson. *The History of Television, 1942 to 2000*. McFarland & Company, 2003.

[AMZ+06] Thomas Annen, Wojciech Matusik, Matthias Zwicker, Hanspeter Pfister, and Hans-Peter Seidel. Distributed Rendering for Multiview Parallax Displays. In *Proceedings of Stereoscopic Displays and Virtual Reality Systems XIII*, pages 231–240, San Jose, USA, 2006. SPIE Press.

[AOK+10] Jun Arai, Fumio Okano, Masahiro Kawakita, Makoto Okui, Yasuyuki Haino, Makoto Yoshimura, Masato Furuya, and Masahito Sato. Integral Three-Dimensional Television Using a 33-Megapixel Imaging System. *Journal of Display Technology*, 6(10):422–430, October 2010.

[Ash93] Ian Ashdown. Near-Field Photometry: A New Approach. *Journal of the Illuminating Engineering Society*, 22(1):163–180, 1993.

[BBM+01] Chris Buehler, Michael Bosse, Leonard Mcmillan, Steven Gortler, and Michael Cohen. Unstructured Lumigraph Rendering. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 425–432. ACM, 2001.

[BDG+03] Zachary Byers, Michael Dixon, Kevin Goodier, Cindy M. Grimm, and William D. Smart. An autonomous robot photographer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3 of *IROS*, pages 2636 – 2641 vol.3, October 2003.

[BFH+04] Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan. Brook for GPUs: Stream Computing on Graphics Hardware. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 777–786, New York, USA, 2004. ACM.

[BJK08] Bogumil Bartczak, Daniel Jung, and Reinhard Koch. Real-Time Neighborhood Based Disparity Estimation Incorporating Temporal Evidence. In *Proceedings of the Deutsche Arbeitsgemeinschaft für Mustererkennung*, DAGM-Symposium, pages 153–162, 2008.

[BKB07] Tibor Balogh, Péter T. Kovács, and Attila Barsi. Holovizio 3D Display System. In *3D Television Conference*, 3DTV, pages 1–4, May 2007.

[BN76] James F. Blinn and Martin E. Newell. Texture and Reflection in Computer Generated Images. *Communications of the ACM*, 19(10):542–547, October 1976.

[But06] Jeremy G. Butler. *Television: Critical Methods and Applications*. Routledge Communication Series. Taylor & Francis, 2006.

[BWSF06] Carsten Benthin, Ingo Wald, Michael Scherbaum, and Heiko Friedrich. Ray Tracing on the CELL Processor. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing*, pages 15–23, 2006.

[CFS02] Wagner T. Corrêa, Shachar Fleishman, and Cláudio T. Silva. Towards Point-Based Acquisition and Rendering of Large Real-World Environments. In *Proceedings of the 15th Brazilian*

Symposium on Computer Graphics and Image Processing, SIBGRAPI, page 59, Washington, DC, USA, 2002. IEEE Computer Society.

[CHH02] Nathan A. Carr, Jesse D. Hall, and John C. Hart. The Ray Engine. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS, pages 37–46, Aire-la-Ville, Switzerland, 2002. Eurographics Association.

[CLPP60] Louis J. Cutrona, Emmett N. Leith, Carmen J. Palermo, and Leonard J. Porcello. Optical data processing and filtering systems. *IRE Transactions on Information Theory*, 6(3):386–400, 1960.

[CNSD93] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 135–142, 1993.

[CS09] Chong Chen and Dan Schonfeld. Geometrical plenoptic sampling. In *16th IEEE International Conference on Image Processing*, ICIP, pages 3769 –3772, November 2009.

[CTCS00] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 307–318, New York, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Cur23] John S. Curwen. Advertising Device or Toy, United States patent US 1,475,430, November 1923.

[CW93] Shenchang E. Chen and Lance Williams. View Interpolation for Image Synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 279–288, New York, USA, 1993. ACM.

[DEG+12] Tomáš Davidovič, Thomas Engelhardt, Iliyan Georgiev, Philipp Slusallek, and Carsten Dachsbacher. 3D Rasterization: A

Bridge between Rasterization and Ray Casting. *Proceedings of the Graphics Interface Conference*, pages 201 – 208, 2012.

[DS07] Andreas Dietrich and Philipp Slusallek. Adaptive Spatial Sample Caching. In *Proceedings of the IEEE/EG Symposium on Interactive Ray Tracing*, pages 141–147, September 2007.

[DSS06] Andreas Dietrich, Jörg Schmittler, and Philipp Slusallek. World-Space Sample Caching for Efficient Ray Tracing of Highly Complex Scenes. Technical Report TR-2006-01, Computer Graphics Group, Saarland University, 2006.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 11–20. ACM, 1996.

[Duc79] Claude E. Duchon. Lanczos Filtering in One and Two Dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, August 1979.

[DWBS03] Andreas Dietrich, Ingo Wald, Carsten Benthin, and Philipp Slusallek. The OpenRT Application Programming Interface – Towards A Common API for Interactive Ray Tracing. In *Proceedings of the OpenSG Symposium*, pages 23–31, Darmstadt, Germany, 2003. Eurographics Association.

[DYB98] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *Eurographics Rendering Workshop*, pages 105–116, 1998.

[ES08] Jan-Friso Evers-Senne. *Plenoptic modelling and rendering of complex rigid scenes.* PhD thesis, 2008.

[ESK05] Jan-Friso Evers-Senne and Reinhard Koch. Image-based rendering of complex scenes from a multi-camera rig. *IEEE Proceedings of Vision, Image and Signal Processing*, 152(4):470–480, August 2005.

[Fav05] Gregg E. Favalora. Volumetric 3D Displays and Application Infrastructure. *Computer*, 38(8):37–44, 2005.

[FB74] Raphael A. Finkel and Jon L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974. 10.1007/BF00288933.

[FCOL99] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic Camera Placement for Image-Based Modeling. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, PG, pages 12–20, Washington, DC, USA, 1999. IEEE Computer Society.

[FWL+11] Miquel Farre, Oliver Wang, Manuel Lang, Nikolce Stefanoski, Alexander Hornung, and Aljoscha Smolic. Automatic content creation for multiview autostereoscopic displays using image domain warping. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, ICME, pages 1–6, Washington, DC, USA, 2011. IEEE Computer Society.

[GD98] Jeffrey P. Grossman and William J. Dally. Point Sample Rendering. In *Rendering Techniques*, pages 181–192. Springer, 1998.

[Ger36] Arun Gershun. The Light Field. *Journal of Mathematics and Physics*, XVIII:51–151, 1936. translated by P. Moon and G. Timoshenko.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 43–54. ACM, 1996.

[GKPS12] Iliyan Georgiev, Jaroslav Křivánek, Stefan Popov, and Philipp Slusallek. Importance Caching for Complex Illumination. *Computer Graphics Forum*, 31(2), 2012. EUROGRAPHICS.

[Gou71] Henri Gouraud. Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, 1971.

[GRHS08] Iliyan Georgiev, Dmitri Rubinstein, Hilko Hoffmann, and Philipp Slusallek. Real Time Ray Tracing on Many-Core-Hardware. In *Proceedings of the 5th INTUITION Conference on Virtual Reality*, October 2008.

[Hal94] Michael W. Halle. Holographic Stereograms as Discrete Imaging Systems. In *Proceedings SPIE Practical Holography*, pages 73–84, 1994.

[HB41] Alfred H. Holway and Edwin G. Boring. Determinants of Apparent Visual Size with Distance Variant. *The American Journal of Psychology*, 54(1):pp. 21–37, 1941.

[HB97] Donald Hearn and M. Pauline Baker. *Computer Graphics, C Version (2nd Edition)*. Prentice Hall, 1997.

[HC02] Mark E. Holzbach and David T. Chen. Rendering methods for full parallax autostereoscopic displays, United States patent US 6,366,370, April 2002.

[HDFP11] Nicolas S. Holliman, Neil A. Dodgson, Gregg E. Favalora, and Lachlan Pockett. Three-Dimensional Displays: A Review and Applications Analysis. *IEEE Transactions on Broadcasting*, 57(2):362 –371, June 2011.

[Hec86] Paul S. Heckbert. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.

[HHN+02] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner, and James T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 693–702, New York, USA, 2002. ACM.

[HK97] Michael W. Halle and Adam B. Kropp. Fast Computer Graphics Rendering for Full Parallax Spatial Displays. *Proceedings of SPIE*, Vol. 3011:p. 105–112, 1997.

[HKP+99] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc J. Van Gool. Plenoptic Modeling and Rendering from

Image Sequences Taken by Hand-Held Camera. In *Proceedings of the 21st Deutsche Arbeitsgemeinschaft für Mustererkennung*, DAGM-Symposium, pages 94–101, London, UK, 1999. Springer-Verlag.

[HWRH13] Felix Heide, Gordon Wetzstein, Ramesh Raskar, and Wolfgang Heidrich. Adaptive Image Synthesis for Compressive Displays. *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, 32(4):1–11, 2013. SIGGRAPH.

[HZ00] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[Ive03] Frederic E. Ives. Parallax Stereogram and Process of Making Same, United States patent US 725,567, April 1903.

[JJ05] R. Barry Johnson and Gary A. Jacobsen. Advances in lenticular lens arrays for visual display. In *Proceedings of SPIE*, volume 5874, pages 587406–11, 2005.

[JK10] Daniel Jung and Reinhard Koch. Efficient Depth-Compensated Interpolation for Full Parallax Displays. In *5th International Symposium on 3D Data Processing, Visualization and Transmission*, 3DPVT, 2010.

[JK11a] Daniel Jung and Reinhard Koch. A Best-Next-View-Selection Algorithm for Multi-View Rendering. In *3D Imaging, Modeling, Processing, Visualization and Transmission*, 3DIMPVT, 2011.

[JK11b] Daniel Jung and Reinhard Koch. Efficient Rendering of Light Field Images. In Daniel Cremers, Marcus Magnor, Martin R. Oswald, and Lihi Zelnik-Manor, editors, *Video Processing and Computational Video*, volume 7082 of *Lecture Notes in Computer Science*, pages 184–211. Springer Berlin Heidelberg, 2011.

[JK13] Daniel Jung and Reinhard Koch. Image Based Rendering from Perspective and Orthographic Images for Autostereoscopic Multi-View Displays. In *International Workshop on Vision, Modeling, and Visualization*, VMV, pages 187–194, 2013.

[JSJK13] Anne Jordt-Sedlazeck, Daniel Jung, and Reinhard Koch. Refractive Plane Sweep for Underwater Images. In *German Conference on Pattern Recognition*, GCPR, pages 333–342, 2013.

[KCK09] Maik Keller, Nicolas Cuntz, and Andreas Kolb. Interactive Dynamic Volume Trees on the GPU. In *14th International Workshop on Vision, Modeling, and Visualization*, VMV, November 2009.

[KLL+13] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proceedings of Joint 3DIM/3DPVT Conference*, 3DV, page 8, June 2013. accepted for oral presentation.

[KS98] Kiriakos N. Kutulakos and Steven M. Seitz. What Do N Photographs Tell Us about 3D Shape? Technical report, 1998.

[Lev44] Kenneth Levenberg. A Method for the Solution of Certain Non-linear Problems in Least-Squares. *Quarterly of Applied Mathematics*, 2:164–168, July 1944.

[Lev71] Robert E. Levin. Photometric Characteristics of Light Controlling Apparatus. *Illuminating Engineering, Vol. 66, No. 4, pp. 205-215*, Vol. 66(No. 4):pp. 205–215, 1971.

[Lev06] Marc Levoy. Light Fields and Computational Imaging. *Computer*, 39(8):46–55, 2006.

[LH96] Marc Levoy and Pat Hanrahan. Light Field Rendering. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 31–42. ACM, 1996.

[LIE98] Martin S. Leung, Neil A. Ives, and Genghmun Eng. Three-Dimensional Real-Image Volumetric Display System and Method, United States patent US 5,745,197, April 1998.

[Lin68] David C. Lindberg. The theory of Pinhole images from antiquity to the thirteenth century. *Archive for History of Exact Sciences*, 5(2):154–176, 1968.

[Lip08] Gabriel Lippmann. Epreuves réversibles. Photographies intégrales. *Comptes Rendus De l'Académie Des Sciences De Paris*, 146:446–451, 1908.

[Low06] Kok-Lim Low. *View planning for range acquisition of indoor environments*. PhD thesis, Chapel Hill, NC, USA, 2006. AAI3207396.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 131–144, New York, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[LR98] Dani Lischinski and Ari Rappoport. Image-Based Rendering for Non-Diffuse Synthetic Scenes. In *Rendering techniques: proceedings of the Eurographics Workshop in Vienna*, pages 301–314, 1998.

[LS00] Zhouchen Lin and Heung-Yeung Shum. On the number of samples needed in light field rendering with constant-depth assumption. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1 of *CVPR*, pages 588 –595 vol.1, 2000.

[LWH+12] Douglas Lanman, Gordon Wetzstein, Matthew Hirsch, Wolfgang Heidrich, and Ramesh Raskar. Beyond Parallax Barriers: Applying Formal Optimization Methods to Multi-Layer Automultiscopic Displays. *Proceedings SPIE 8288, Stereoscopic Displays and Applications XXIII*, pages 82880A–82880A–13, 2012.

[Mar63] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. In *Proceedings of the Annual*

Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pages 39–46. ACM, 1995.

[MF98] Nikolaos A. Massios and Robert B. Fisher. A Best Next View Selection Algorithm Incorporating a Quality Criterion. In *British Machine Vision Conference*, BMVC, 1998.

[MGAK03] William R. Mark, R. Steven Glanville, Kurt Akeley, and Mark J. Kilgard. Cg: A System for Programming Graphics Hardware in a C-like Language. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 896–907, New York, USA, 2003. ACM.

[Mis11] Tomoyuki Mishina. Three-dimensional television system based on integral photography. In *IEEE Visual Communications and Image Processing*, VCIP, pages 1–4, November 2011.

[MMSW06] Philipp Merkle, Karsten Müller, Aljoscha Smolic, and Thomas Wiegand. Efficient Compression of Multi-View Video Exploiting Inter-View Dependencies Based on H.264/MPEG4-AVC. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, ICME, pages 1717–1720, july 2006.

[MMW10] Philipp Merkle, Karsten Müller, and Thomas Wiegand. 3D video: acquisition, coding, and display. *IEEE Transactions on Consumer Electronics*, 56(2):946–950, 2010.

[MP04] Wojciech Matusik and Hanspeter Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 814–824, New York, USA, 2004. ACM.

[MRG03] Marcus Magnor, Prashant Ramanathan, and Bernd Girod. Multi-View Coding for Image-based Rendering using 3-D Scene Geometry. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1092–1106, November 2003.

[MS81] Parry Moon and Domina E. Spencer. *The Photic Field*. MIT Press, 1981.

[MSD+08] Karsten Müller, Aljoscha Smolic, Kristina Dix, Peter Kauff, and Thomas Wiegand. Reliability-based generation and view synthesis in layered depth video. In *IEEE 10th Workshop on Multimedia Signal Processing*, pages 34–39, 2008.

[NBGS08] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable Parallel Programming with CUDA. *Queue*, 6(2):40–53, March 2008.

[Oko76] Takanori Okoshi. *Three-Dimensional Imaging Techniques*. Academic Press, Inc. (London) LTD., 1976.

[OLG+07] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray Tracing on Programmable Graphics Hardware. *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, 21(3):703–712, July 2002. ISSN 0730-0301, SIGGRAPH.

[PCD+97] Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle. View-Based Rendering: Visualizing Real Objects From Scanned Range and Color Data. In Julie Dorsey and Phillipp Slusallek, editors, *Proceedings of the Eighth Eurographics Workshop on Rendering, Rendering Techniques*, pages 23–34, New York, USA, 1997. Springer Wien.

[PGSD13] Stefan Popov, Iliyan Georgiev, Philipp Slusallek, and Carsten Dachsbacher. Adaptive Quantization Visibility Caching. *Computer Graphics Forum*, 32(2), 2013. EUROGRAPHICS.

[Pho75] Bui T. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[PLK01] Andrey N. Putilin, Andrew A. Lukianitsa, and K. Kanashin. Stereodisplay with neural network image processing. In *Proceedings of SPIE*, volume 4511, pages 245–250, 2001.

[RL00] Szymon Rusinkiewicz and Marc Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 343–352, July 2000.

[Rol53] Wilhelm Rollmann. Zwei neue stereoskopische Methoden. *Annalen der Physik*, 166(9):186–187, 1853.

[Ros05] Randi J. Rost. *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional, 2005.

[SD96] Steven M. Seitz and Charles R. Dyer. View Morphing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 21–30, New York, USA, 1996. ACM.

[SD97] Steven M. Seitz and Charles R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pages 1067–1073, June 1997.

[SGHS98] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 231–242, New York, USA, 1998. ACM.

[SH99] Heung-Yeung Shum and Li-Wei He. Rendering with Concentric Mosaics. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 299–306, New York, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 245–254, New York, USA, 1985. ACM.

[SHS99] Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. Adaptive Acquisition of Lumigraphs from Synthetic

Scenes. In *Computer Graphics Forum*, volume 18 of *Eurographics*. Blackwell Publishers Ltd, 1999.

[SK99] Heung-Yeung Shum and Sing B. Kang. A survey of image-based rendering techniques. In *SPIE Videometrics*, pages 2–16, 1999.

[SRR03] William R. Scott, Gerhard Roth, and Jean-François Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Comput. Surv.*, 35:64–96, March 2003.

[SS97] Richard Szeliski and Heung-Yeung Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 251–258, New York, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[SS98] Heung-Yeung Shum and Richard Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *Sixth International Conference on Computer Vision*, pages 953–956, January 1998.

[SSM09] Danilo Schneider, Ellen Schwalbe, and Hans-Gerd Maas. Validation of geometric models for fisheye lenses. *Journal of Photogrammetry and Remote Sensing*, 64(3):259 – 266, 2009. Theme Issue: Image Analysis and Image Engineering in Close Range Photogrammetry.

[SWR+08] Jacob Ström, Per Wennersten, Jim Rasmusson, Jon Hasselgren, Jacob Munkberg, Petrik Clarberg, and Tomas Akenine-Möller. Floating-point Buffer Compression in a Unified Codec Architecture. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH, pages 75–84, Aire-la-Ville, Switzerland, 2008. Eurographics Association.

[SWW+04] Jörg Schmittler, Sven Woop, Daniel Wagner, Wolfgang J. Paul, and Philipp Slusallek. Realtime Ray Tracing of Dynamic Scenes on an FPGA Chip. In *Proceedings of Graphics Hardware*, pages 95–106, 2004.

[Sze93] Richard Szeliski. Rapid octree construction from image sequences. *Proceedings of Computer Vision, Graphics, and Image Processing*, 58(1):23–32, 1993. CVGIP.

[TAT95] Konstantinos A. Tarabanis, Peter K. Allen, and Roger Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86 –104, February 1995.

[Tel98] Seth Teller. Toward Urban Model Acquisition from Geo-Located Images. In *Proceedings of the 6th Pacific Conference on Computer Graphics and Applications*, PG, page 45, Washington, DC, USA, 1998. IEEE Computer Society.

[TLRS+08] Severin Todt, Matthias Langer, Christof Rezk-Salama, Andreas Kolb, and Klaus-Dieter Kuhnert. Spherical Light Field Rendering in Application for Analysis by Synthesis. *Int. J. on Intell. Systems and Techn. and App. Issue on Dynamic 3D Imaging*, 5(1):304 – 314, 2008. IJISTA.

[vdBLF96] Christian J. van den Branden Lambrecht and Joyce E. Farrell. Perceptual Quality Metric for Digitally Coded Color Images. In *Images, Proceedings of the European Signal Processing Conference*, pages 10–13, 1996.

[WAA+00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 287–296, New York, USA, 2000. ACM.

[WB89] George Wolberg and Terrance E. Boult. Separable Image Warping with Spatial Lookup Tables. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 369–378, New York, USA, 1989. ACM.

[WBSS04] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[WDA99] Laurana M. Wong, Christophe Dumont, and Mongi A. Abidi. Next best view system in a 3D object modeling task. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, CIRA, pages 306 –311, 1999.

[WDS04] Ingo Wald, Andreas Dietrich, and Philipp Slusallek. An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models. In *Proceedings of the Eurographics Symposium on Rendering, Rendering Techniques*, pages 81–92, June 2004.

[Web85] Leonhard Weber. Intensitätsmessungen des diffusen Tageslichtes. *Annalen der Physik*, 262(11):374–389, 1885.

[Wes90] Lee Westover. Footprint Evaluation for Volume Rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 367–376, New York, USA, 1990. ACM.

[Wet11] Gordon Wetzstein. *Computational Plenoptic Image Acquisition and Display*. PhD thesis, Department of Computer Science, University of British Columbia, 2011.

[Whe38] Charles Wheatstone. Contributions to the Physiology of Vision. Part the First. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision. *Philosophical Transactions of the Royal Society of London*, 128:371–394, 1838.

[WHG84] Hank Weghorst, Gary Hooper, and Donald P. Greenberg. Improved Computational Methods for Ray Tracing. *ACM Transactions on Graphics*, 3(1):52–69, January 1984.

[WHLP96] Tomas Werner, Vaclav Hlavac, Ales Leonardis, and Tomas Pajdla. Selection of Reference Views for Image-Based Representation. *International Conference on Pattern Recognition*, 1:73–77, 1996.

[WM08] Stefan Winkler and Praveen Mohandas. The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. *IEEE Transactions on Broadcasting*, 54(3):660–668, 2008.

[Wol90] George Wolberg. *Digital Image Warping*, volume 10662. IEEE computer society press California, 1990.

[WSBW01] Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive Rendering with Coherent Ray Tracing. *Computer Graphics Forum*, 20(3):153–165, 2001.

[WZ63] Hans Wallach and Carl Zuckerman. The Constancy of Stereoscopic Depth. *The American Journal of Psychology*, 76(3):pp. 404–412, 1963.

[XPLL11] Xuewu Xu, Yuechao Pan, Phyu P.M.Y. Lwin, and Xinan Liang. 3D holographic display and its data transmission requirement. In *International Conference on Information Photonics and Optical Communications*, IPOC, pages 1–4, October 2011.

[YKO10] Fahri Yaraş, Hoonjong Kang, and Levent Onural. State of the Art in Holographic Displays: A Survey. *Journal of Display Technology*, 6(10):443–454, October 2010.

[ZC03a] Cha Zhang and Tsuhan Chen. A system for active image-based rendering. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1 of *ICME*, pages I – 233–6 vol.1, July 2003.

[ZC03b] Cha Zhang and Tsuhan Chen. Non-uniform sampling of image-based rendering data with the position-interval error (PIE) function. In *Visual Communication and Image Processing*, VCIP, Lugano, Switzerland, July 2003.

[ZHWG08] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time KD-tree Construction on Graphics Hardware. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques Asia*, SIGGRAPH Asia, pages 126:1–126:11, New York, USA, 2008. ACM.

[ZMD+06] Matthias Zwicker, Wojciech Matusik, Frédo Durand, Hanspeter Pfister, and Clifton Forlines. Antialiasing for automultiscopic 3D displays. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques, Sketches*, SIGGRAPH, New York, USA, 2006. ACM.

[ZPvBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface Splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 371–378, New York, USA, 2001. ACM.

Glossary

A	mapping of view to spherical camera coordinate angles	19, 34, 35
B	mapping of a lens system to its 3D position	20, 34, 35
$\flat$	line connecting two camera centres (baseline)	51–54, 144, 145
$b_{\dashv}$	frustum bottom	28–30, 42, 55
C	camera coordinate system	22, 23, 27, 51–53, 55, 56, 58, 61, 62, 70, 79, 92, 93, 95, 137, 138, 140, 141, 148–151, 153, 156, 170, 174
$\mathbf{c}$	centre of an image in image coordinates	31, 181
c	number of channels of an image	21, 36, 37, 49
d	depth, usually measured as the Euclidean distance to the optical centre	8, 54, 144, 145
δ	disparity	9, 52–55, 144
δ_d	depth range used for blending	75
D	depth or disparity image	21, 55, 56, 58, 74, 75, 132
d_f	distance between neighbouring orthographic views	180
d_{lp}	lens pitch	18, 36, 174
$d_{\mathcal{OP}}$	distance to the virtual display plane, where the interpolated orthographic and perspective images are exchanged	145

d_p	the aperture of a lens system relates to the size of one pixel	174
d_z	depth, measured as the Z-component of the camera's coordinate system	41, 53–55, 120, 122, 123
E	mapping of view to axis angles	55
$\mathbf{E}$	normalised direction to observer	170
f	far clipping plane	28–30, 33, 42, 54, 172
f_a	attracting force	180, 181
f_r	repelling force	180
$\boldsymbol{\gamma}$	ray direction	32
H	set of viewing directions (angle to axis)	54, 55
$\mathbb{I}$	the set of up to four input views for interpolation	73, 75, 87
I	colour image	21, 28, 34, 36, 37, 55, 58, 73–75, 87, 88, 123, 124, 132, 212
$\mathcal{I}_c$	combined interpolation of perspective and orthographic images	152, 153, 157–160
$\mathcal{I}_o$	interpolation of orthographic images	153, 156–158
$\mathcal{I}_p$	interpolation of perspective images	152, 153, 157, 158
$\mathbb{J}$	the set of blended views	74, 75, 88
$\overleftrightarrow{\mathbb{K}}^i_{u,v}$	set of image coordinates around the 3×3 region of (u, v) of image i	56, 88
λ	wavelength	18, 20, 33
$l_{\dashv}$	frustum left	28–30, 42, 55
LF	light field	20, 34
$\mathbf{L}$	normalised direction to light source	170

$\mathcal{M}$	modelling transformation	23
n	near clipping plane	27–30, 33, 42, 52–54, 149, 172
$\mathbf{n}^{2D}$	normalised 2D vector	32
$\mathbf{n}$	surface normal	170
ν	number of horizontal views of a multi-view display	12, 54
O	object coordinate system	23, 24, 61, 80, 149, 150
$\mathcal{O}$	orthographic projection	24, 30, 34, 42, 145, 146, 209
$\mathcal{P}$	perspective projection	24, 28, 34, 145, 146, 209
φ	polar angle	18–20, 30, 32–35, 104, 172
Π_e	epipolar plane	51
Π_I	image plane	27–29
P	plenoptic function	18, 20, 33–35, 41
$\boldsymbol{p}$	point in Euclidean 3-space	8, 22, 32, 33, 51–54, 61, 62, 119, 120, 123, 124, 170
Ψ	maximum viewing angle	12, 19
ψ	FOV	31, 32, 41, 53, 54, 145, 172, 211
$\psi/2$	Angle Of Field (AOF)	41, 53, 54, 145
$\mathcal{Q}$	transformation of object coordinates into camera coordinates	23, 24
R_a	angular resolution	12, 35
r	radial distance	30–32
$\mathbf{H}$	normalised direction, half between view direction and light direction	170

$r_{\dashv}$	frustum right	28–30, 42, 55
$\mathcal{R}$	rotation	23
$R_{S,R(v)}$	spatial resolution of one view in millimetre per pixel (width, height)	11, 18
R_v	resolution of one view in pixel (width, height)	11, 20, 34, 35, 49, 55, 123
S	function that yields 2D image coordinates, based on a disparity map	56
σ	display fitting value of viewpoint optimization	124
S_v	size of one view in millimetre (width, height)	11, 120, 123
T	all views of a multi-view display	12, 19, 34, 35, 48, 49, 54, 55
θ	azimuthal angle	18–20, 30, 32–35, 104, 172
t	time	18–20, 33
$t_{\dashv}$	frustum top	28–30, 42, 55
$\mathbf{t}$	translation	23
U	width of an image	21, 24, 30–32, 36, 37, 53, 56, 57
Y	set of viewing directions (spherical)	19, 33
V	height of an image	21, 24, 30–32, 36, 37, 53, 56, 57
$\mathbf{V}$	3D position	18–20, 33, 34, 123, 124
$\mathcal{V}$	viewing transformation	22, 23
w_F	scaling factor of the Best-Next-View (BNV) algorithm	124, 125
w_f	weight, based on the distance to the image centre	87–89
w_{I_n}	BNV weight of image I_n	123, 124

$w^i_{L_1}$	weight, based on the L1-norm between target and input image	73–75, 87–89
w_m	weight of the BNV algorithm	124
w_M	minimum weight of the BNV algorithm	124, 125
W	world coordinate system	20, 22, 33
w_p	final weight of the interpolation of perspective images	88, 89
$\mathcal{W}$	viewport transformation	24, 25, 31, 32, 87, 88
w^i_w	weight for blending of the distributed colour information	57, 75, 87–89
Ξ	Frames Per Second (FPS)	49
ξ	number of vertical views of a multi-view display	12, 54
z_n	$\in [0, 1]$ normalised depth	54