BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

REAL-TIME LIGHT FIELD RENDERING ACCELERATION ON GPU

AKCELERACE REAL-TIME LIGHT FIELD RENDERINGU NA GPU

DOCTORAL THESIS
DISERTAČNÍ PRÁCE

AUTHOR **Ing. TOMÁŠ CHLUBNA**
AUTOR PRÁCE

SUPERVISOR **prof. Dr. Ing. PAVEL ZEMČÍK, dr. h. c.**
ŠKOLITEL

BRNO 2024

Abstract

This thesis describes original proposals regarding light field rendering, focusing, compression, streaming, and methods for its optimal usage on 3D displays. Light field rendering is a fast image-based way to render otherwise computationally demanding synthetic or even real-life scenes. An input set of images that captures a scene is used to synthesize a novel view without complex 3D reconstructions. The main scientific contribution of this thesis is a proof of a hypothesis. The hypothesis states that state-of-the-art light field rendering methods can be outperformed by a novel method in terms of visual quality, memory requirements, and time performance. The novel method proposed in this thesis enables the usage of light fields in real time without visual quality degradation and excessive hardware requirements. The hypothesis is proven experimentally and supported by reference implementations and published papers. The thesis also contains other novel proposals that serve as supporting materials for the main method. These proposals address the most crucial issues in light field rendering. The main contribution and other proposals are intended to open a way to the usage of light field assets in the gaming and film industry. This thesis provides an overview of existing state-of-the-art methods, the hypothesis with experimental proof, and a description of proposed applications of light field principles on 3D displays.

Abstrakt

Tato práce popisuje originální návrhy týkající se vykreslování light fieldů, zaostřování, komprese, streamování a metod pro jejich optimální využití na 3D displejích. Vykreslování light fieldů spadá do oblasti tzv. image-based renderingu. Jedná se o rychlý způsob vykreslování jinak výpočetně náročných syntetických nebo také reálných scén. Vstupní sada obrázků, která zachycuje scénu, se používá k syntéze nového pohledu bez složitých 3D rekonstrukcí. Hlavním vědeckým přínosem této práce je důkaz hypotézy. Hypotéza uvádí, že moderní metody vykreslování light fieldů mohou být překonány novou metodou, v oblasti vizuální kvality, paměťových požadavků a časové náročnosti. Nová metoda navržena v této práci umožňuje využití light fieldů v reálném čase bez zhoršení vizuální kvality a bez nadměrných hardwarových požadavků. Hypotéza je experimentálně dokázána a podpořena referenčními implementacemi a publikovanými články. Práce obsahuje i další nové návrhy, které slouží jako podklady pro hlavní metodu. Tyto návrhy řeší zásadní problémy při vykreslování light fieldů. Hlavním přínosem práce je otevření cesty k využití light fieldů v herním a filmovém průmyslu. Tato práce obsahuje přehled stávajících nejmodernějších metod, hypotézu s experimentálním důkazem a popis navrhovaných aplikací light fieldů na 3D displejích.

Keywords

light field, 3D display, GPU, image-based rendering, multiple views, plenoptic function

Klíčová slova

light field, 3D displej, GPU, image-based rendering, více pohledů, plenoptická funkce

Reference

CHLUBNA, Tomáš. *Real-Time Light Field Rendering Acceleration on GPU.* Brno, 2024. Doctoral thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Dr. Ing. Pavel Zemčík, dr. h. c.

Rozšířený abstrakt

Light field rendering patří mezi metody tzv. image-based renderingu. Tyto metody jsou určeny k vykreslování nových pohledů na 2D či 3D scény pouze na základě obrazového vstupu. Vstupem těchto metod jsou pouze obrázky zachycující danou scénu z různých úhlů pohledu. Není potřeba např. popis geometrie, nastavení materiálů povrchů nebo rozmístění světel ve scéně. Pojem light field (světelné pole) je používán pro reprezentaci scény pouze na základě šíření světla. Pro každý bod a každý směr v prostoru je definována tzv. plenoptická funkce. Návratovými hodnotami této funkce jsou vlastnosti světla jako jeho barva či intenzita. Ve výpočetních technologiích je obtížné pracovat s takovou spojitou funkcí, proto je light field definován pouze pro vymezený prostor a diskrétně. V praxi se jedná o množinu 2D obrázků, zpravidla zachycených kamerami umístěnými na přesně definovaných místech. Nejčastěji používané rozmístění kamer je v pravidelné rovinné mřížce.

Použití diskrétního light fieldu pro vykreslení nového pohledu na danou scénu přináší problémy. V oblasti vizuální kvality se jedná o nedostatek informací o scéně. Jelikož obrázky nezachycují spojitou informaci, části scény pro některé nové pohledy musí být doplněny pouze na základě vstupů, a tedy často interpolovány bez hlubší znalosti o reálném obsahu scény. Dále chybějící informace o hloubce ve scéně znemožňuje správné zaostření scény ve všech jejích částech, a scéna je tak částečně rozostřená. V oblastí výkonu je problematická samotná reprezentace diskrétního light fieldu, jelikož nekomprimované obrázky ve vysokém rozlišení zaplní velkou část paměti na grafické kartě. Výše zmíněnou chybějící hloubku ve scéně lze také odhadnout existujícími metodami, nicméně tento odhad je často výpočetně náročný. Existující metody navíc odhadují hloubku pro dané vstupní pohledy, ne však pro zatím neexistující nový pohled, pro nějž je potřeba.

Tato práce vychází ze studia existujících stávajících metod. Na základě těchto metod je stanovena vědecká hypotéza. Tato hypotéza tvrdí, že lze navrhnout novou metodu light field renderingu, která je schopná vykreslit plně zaostřené nové pohledy na scénu, přímo ze vstupních obrázků, bez potřeby dodatečné informace o hloubce, v kvalitě vyšší než u existujících metod, s menšími požadavky na výkon a paměť. Hypotéza je dokázána experimentálně. Důkaz hypotézy je zdokumentován v této práci a vychází z publikovaných vědeckých článků. Hlavním jádrem důkazu je metoda, která na základě vzájemné podobnosti vstupních obrázků vygeneruje tzv. zaostřovací mapu. Tato rastrová mapa v každém pixelu uchovává optimální zaostřovací vzdálenost pro požadovaný nový pohled na scénu. Hodnoty v mapě jsou získány na základě nejmenší chyby, kdy jsou pixely vstupních obrázků porovnávány podle svých barev za účelem vyhledání nejpodobnějších hodnot, a tedy podobných míst ve scéně. Tato mapa se tedy liší od často používané hloubkové či disparitní mapy a nemusí plně odpovídat struktuře scény. Je vypočítána pro nový pohled a definována ve všech místech. Na základě této mapy pak jsou ze vstupních obrázků vybrány potřebné pixely a interpolovány do nového pohledu. Tato metoda je navržena tak, aby plně využila masivního paralelismu na grafických kartách. V rámci výzkumu byla vytvořena referenční implementace a naměřené výsledky odpovídají hypotéze. Metoda je použitelná i pro aplikace vyžadující vykreslení v reálném čase, minimalizuje velikost dat potřebných v paměti a není závislá na často časově náročném strojovém učení.

Mimo hypotézu a důkaz jsou v práci také uvedeny doplňující výsledky výzkumu. Tyto výsledky se zabývají například kompresními metodami vhodnými pro diskrétní light field. Navržená komprese využívá akceleraci na grafické kartě a je kompatibilní s navrženou vykreslovací metodou. Dále se práce zabývá optimálním zobrazením dat na 3D displejích. Toto téma také souvisí s principy používanými v oblasti light fieldu.

Real-Time Light Field Rendering Acceleration on GPU

Declaration

I hereby declare that this thesis was prepared as an original work by the author under the supervision of prof. Dr. Ing. Pavel Zemčík, dr. h. c. All the literary sources, publications and other sources, which were used during the preparation of this thesis were listed.

.....
Tomáš Chlubna
May 30, 2024

Acknowledgements

I would like to thank too many people here. In order not to scare the reader here by the length of this section, I will provide only a very small subset of these people and try to mention the rest indirectly at least. So my huge thanks belong to:

- Prof. Pavel Zemčík for being my supervisor who always found time for me despite his incredibly hectic schedule.
- Tomáš Milet for being almost like my second supervisor who helped me with his incredibly broad knowledge a lot with...basically everything I did, and also for introducing me to the world of K-dramas.
- Q202 - this is not a sci-fi robot but the name of my office where I spent a lot of time talking and laughing...I mean working with my colleagues Michal Vlnas, Matej Karas, Jan Pečiva, Josef Kobrtek, Michal Kula (actually, he is in the neighboring office but he comes a lot to visit), Tomáš Starka, Petr Klepárník, Roman Čižmárik, Tomáš Milet (yea, he deserves to be mentioned twice) and all other people I met there and did not mention. Oh, and thanks to Mr. Rabbit for being a guardian of our office. Also, I am sorry to people at other neighboring offices for us being sometimes too loud.
- David Bařina for letting me work in his team which led to my first scientific publication.
- BUT FIT, this amazing faculty, which gave me all the knowledge in computer science. Although our school is considered quite difficult, I always had a lot of fun here (and got some white hair as well). Now as an employee I still gain a lot of experience in scientific work and also in teaching others. My thanks belong not only to the academics but also to all employees of this faculty for keeping it going. We would be totally lost without our technicians (especially the legendary Mr. Juříček), office workers, and cleaners. You know, IT scientists and real-life practical problems often do not go well together.
- Reviewers of this thesis and all my papers that went through peer review for valuable advice that helped me write better technical texts.
- Mom, well...what else to say? I need to thank her for running the famous Hotel Mama where I still live...since my birth.
- Andrashi for calling me sometimes and talking about any topic possible.
- Mony, Axi, Eddie for listening to me without having a clue what I was talking about.
- Kareru for reminding me that life is difficult. I often tend to forget that.
- Maras for his crazy travel plans.
- My Shadow for not giving up on me and following me all my life.
- To make sure nobody is left out, all people in the world, those I have met, will meet, or never have met or will meet. I am sure everyone would be able to give me something that would make my life nicer. I hope that I could do the same.

Contents

1 Introduction **2**

2 Light Field **3**
2.1 Overview of Light Field 3
2.2 Physical Background 10
2.3 Discrete Approximation 11
2.4 Focusing Methods 14
2.5 Capturing of Discrete Light Field 31
2.6 Compression of Discrete Light Field 33
2.7 GPU-Accelerated Decompression of Discrete Light Field 38
2.8 Decompression Evaluation 46

3 Hypothesis & Experimental Proof **58**
3.1 Hypothesis 58
3.2 Real-time Per-Pixel Focusing 59
3.3 Improvements of the Focusing Algorithm 63
3.4 Experimental Evaluation with Standard Datasets 72
3.5 Focusing Performance Optimizations 82
3.6 Experimental Evaluation with Novel 4K Dataset 85
3.7 Novel Dataset Capturing Method 92
3.8 Hypothesis Proof Summary 96

4 Light Field Applications and 3D Displays **99**
4.1 3D Displays 99
4.2 3D-Display-Friendly Frames Extraction 106
4.3 Focusing Artifacts Mitigation 126
4.4 Capturing Camera Trajectory Distortion 140

5 Conclusion **154**

Bibliography **155**

Chapter 1
Introduction

A 3D scene in computer graphics is often described by its geometry, materials, and light sources. Rendering techniques, such as path tracing or rasterization, sample the scene, obtain the necessary visual information, and produce a novel view. However, these methods can be performance-demanding and the 3D representation of the scene might not be easy to acquire. To render a view of a scene, only a light-propagation model suffices. This model, called the *light field*, describes the incoming light for any point and direction in the scene. Lightweight image-based rendering methods can be used to produce novel views. Scenes can be captured by light field cameras, camera arrays, or a moving camera. The captured light field consists of multiple views of the scene. Synthetic or even real-life scenes can be easily captured and reconstructed.

The main advantage of light field, in comparison to standard photography, is the option of advanced editing in post-process. Standard photos and videos are captured from one position and with one focus distance and aperture settings at a time. Light field can be viewed from different angles or focused at different parts of the scene. A virtual camera motion in a 3D environment can be simulated based only on the input set of 2D images. Such editing and viewing options are in this thesis grouped to the term *light field rendering*.

Light fields are currently not widely used in industry. One drawback is the sampling rate. A discrete approximation of light field, in the form of 2D images, does not contain continuous information about the scene. Artifacts appear in parts of the scene with missing information. Another issue is that storing the images requires a lot of memory. Also, to achieve a completely sharp novel image, the geometry of the scene needs to be estimated. Preprocessing of the data is not always possible in real-time applications. Depth or geometric input data increase the memory requirements and might not always be available. One inherent limitation of light field is the impossibility to use standard 3D operations, such as collision detection or physics simulation. The methods proposed in this thesis aim to solve the issues related to quality and performance.

Based on the study of existing light field rendering methods, the central hypothesis of this thesis is stated. According to the hypothesis, a novel light field rendering method can be proposed and implemented. The advantages of this method over the state of the art include a better visual quality of the result, less input data, and less memory and computational requirements. The proposed method is described in this thesis and the hypothesis is experimentally proven.

Light field rendering can be utilized to produce an input for virtual reality headsets, 3D autostereoscopic displays, 3D tablets, or standard 2D screens with simulated 3D movement. Part of this thesis explores the 3D displays and proposes methods for optimal retrieval and processing of the input light field data. In summary, this thesis contains original proposals covering the whole light field workflow, from the capturing of the data, their compression, streaming to the GPU, rendering of the novel view, to the potential application in industry.

Chapter 2
Light Field

This chapter contains general and in-depth explanations on the topic of light field. The basic principles of light field rendering and related topics, such as acquisition of the data and their compression, are explained in this chapter. The description of light field state-of-the-art methods serves as a theoretical background for the novel proposals in the later parts of this thesis.

2.1 Overview of Light Field

As the name *light field* suggests, a field of light can be defined for a given scene. Such a field would contain information about the incoming light at any point in the scene; see Fig. 2.1. This universal definition does not involve working with materials, surfaces, or geometry. However, it is difficult and impractical to work with such general definitions. In computer graphics, the light field is defined only on a subset of space. Also, instead of an ideal theoretical continuous definition, a discrete approximation is used. The scene is described as static, and the light is defined only for specific wavelengths corresponding to the human visual system. In practice, a light field can be used as an alternate representation of a 3D scene, different from the classic structural boundary or volumetric representation; see Fig. 2.2.

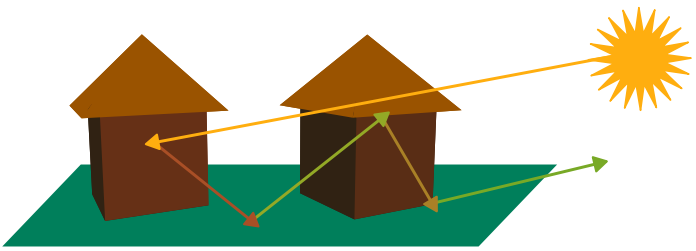
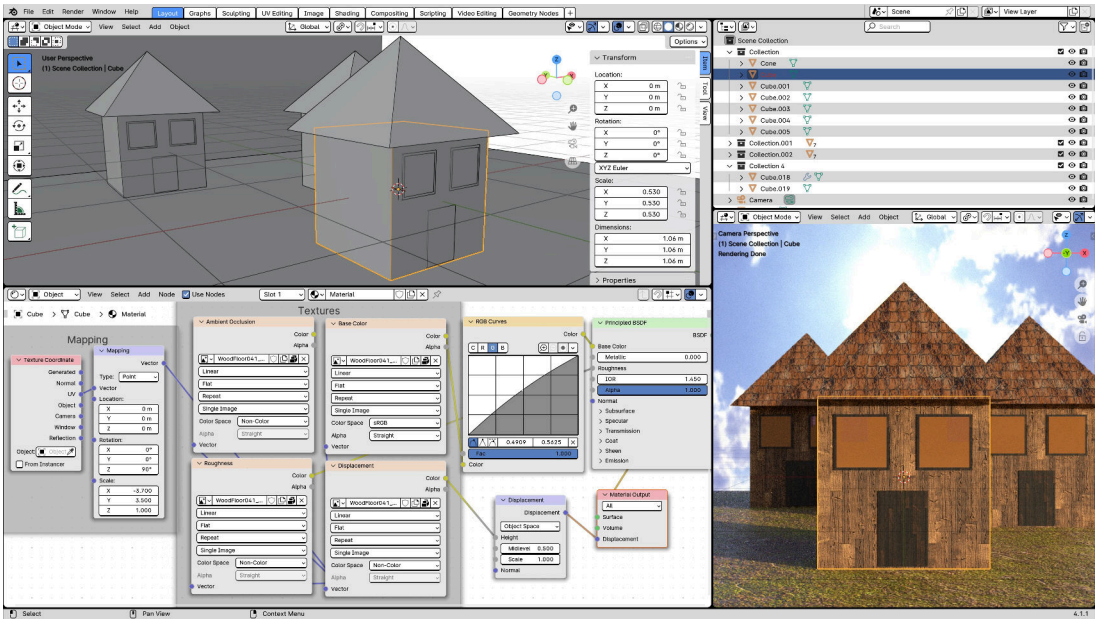
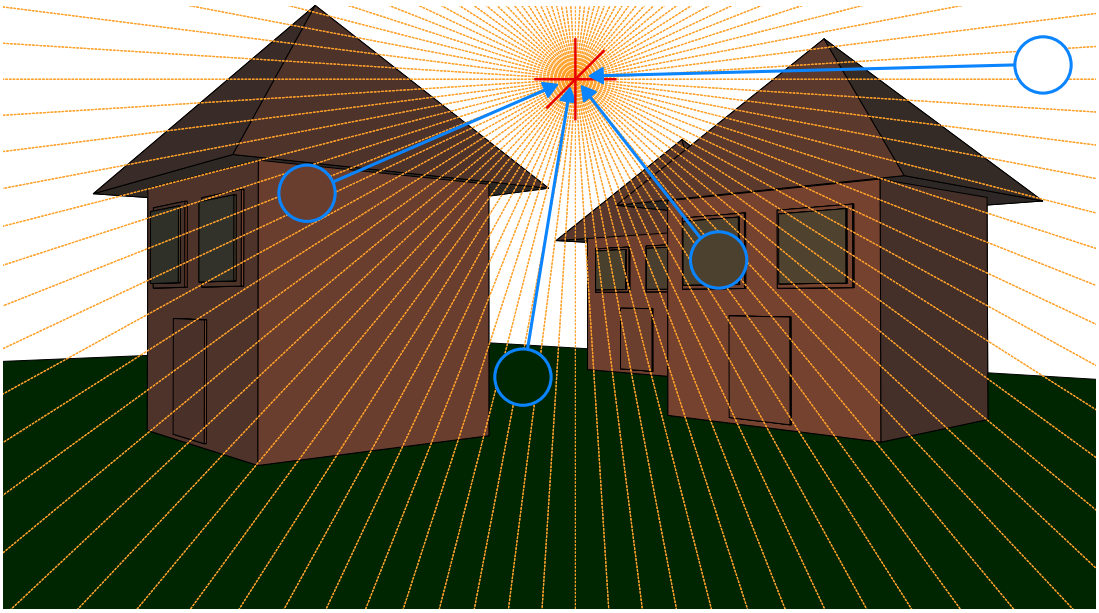


Figure 2.1: Ray coming from a source changes its color according to the materials contacted.

Light field contains visual information for a defined position and a direction in space. Structural representations describe the scene using information about its surface or volume attributes, such as definition of the shape of objects, textures, lights, or material settings [91]. Light field is usually represented by a set of images that contain differently positioned views of the same scene. Fig. 2.3 demonstrates the standard light field rendering pipeline. The scene is first captured in multiple views from defined camera positions. The images are aligned and passed to the light field rendering method. The rendering method selects the necessary images and samples their pixels that are used in the synthesis of the novel view.



(a) 3D editor Blender shows a geometry visualization, material editor, rendered view, etc.



(b) One sampling point with all possible rays are indicated, and four example rays with incoming light colors in the circles are highlighted in the theoretical light field representation.

(c) Practical light field approximation consists of a set of images.

Figure 2.2: 3D scene can be represented by a definition of its geometry and materials. A window in the Blender software depicts such a representation. A field of light can be defined for any point, any ray, from any direction. In practice, such a representation is approximated by a set of images that capture the scene from different positions.

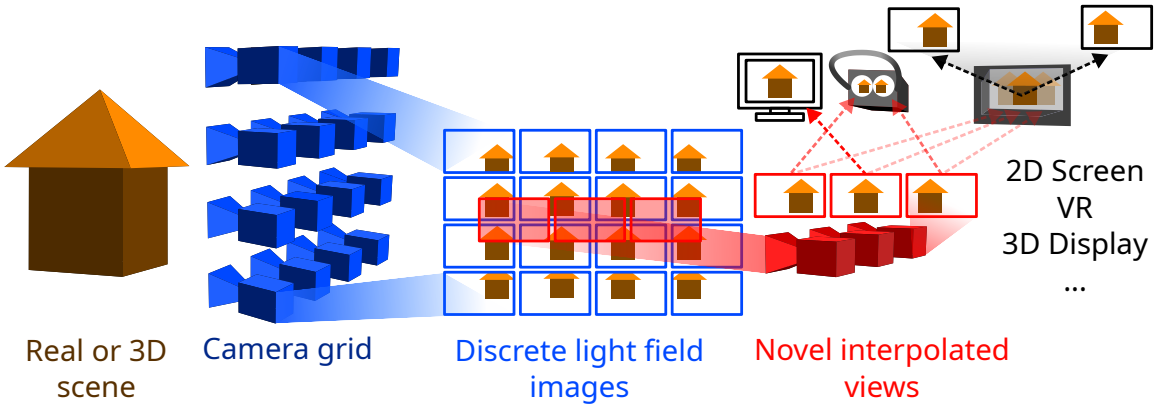


Figure 2.3: A grid of cameras captures a synthetic or real scene. A set of images, containing various views of the scene, is stored or streamed directly to the rendering method. Novel views are generated using light field rendering methods. These views can be used in 2D and 3D display devices to create a 3D experience in real time.

2.1.1 Practical Highlights of Light Field

Light field representation of the scene is suitable for rendering purposes. The advantages of light field rendering methods are mainly the constant, content-independent rendering time, and straightforward capture of a synthetic or real-life scene. The drawback of the light field is that it does not implicitly contain all necessary information about the scene geometry for advanced interaction with the scene, such as physics simulation or collision detection.

A quick motivational experiment was conducted to show how light field rendering compares to path-tracing and rasterization; see Tab. 2.1. NVIDIA OptiX render and denoising with adaptive sampling in Cycles engine and rasterization (Eevee engine) in a 3D modeling and rendering software Blender 3.6 was compared with a light field rendering method [10]. The measurements were performed on NVIDIA® GeForce RTX™ 2070.

	Used GPU memory [MB]			Render time				PSNR [dB]		
Scene	**PT**	**Raster**	**LF**	**PT**	**PT 1s**	**Raster**	**LF**	**PT 1s**	**Raster**	**LF**
Cornell	1698	811	1200	172.01s	2.73s	0.20s	0.0512s	28.40	11.66	43.67
Bonfire	2230	1084		163.97s	4.62s	0.55s		35.05	17.11	38.40
Low	2398	1736		45.41s	3.07s	2.48s		34.76	21.55	38.76
Simple	2298	1159		12.27s	3.01s	2.27s		41.20	31.48	39.28

Table 2.1: The table contains a comparison of path tracing, rasterization, and light field rendering. Light field requires constant memory footprint comparable with other techniques but offers up to 60× faster visualization to path tracing with one sample and 20× faster to rasterization. Four different scenes from a 4K light field dataset [10] were compared. **PT** refers to path tracing, **PT 1s** to path-tracing with one sample and denoising, **Raster** to rasterization and **LF** to light field rendering. **PT** was used as a reference for the PSNR measurement.

The results depend on the type of the scene. Note that path tracing can be used to generate scenes for light field rendering. From a theoretical point of view, path-tracing can be viewed as an algorithm that populates a discrete subset of the light field by probing the ray directions defined by the camera parameters. The experiment is focused on the rendering of the novel view, based on existing light field dataset. The production of the

dataset itself is not relevant for this motivational measurement. The main focus of this experiment is to evaluate the options for real-time rendering. This experiment demonstrates how light field rendering might be a better choice than a standard rendering implemented in the popular 3D rendering software and potentially than many other implementations.

The results show that light field rendering does not require as much memory as path tracing. Rasterization can require less memory when only simple geometry is present without a lot of textures. Light field rendering offers a significant rendering speedup compared to the other approaches. Although rasterization is commonly used in real-time graphics, a lot of effects need to be simulated and are not possible to physically compute. The speed of light field rendering depends solely on the number of necessary texture read operations, which would be five per pixel for rendering with precomputed focus distances. Note that the rasterization implementation in Blender might not be optimal and the times may be faster, but the visual quality of the result would still be lacking due to its inability to fully simulate path-tracing results. This is also the reason why the GPU industry has focused on GPU-accelerated ray tracing in recent years [229]. Path tracing with only one sample and denoising is still worse in both speed and visual quality compared to light field rendering. Generally, the usability of light fields increases with higher complexity of the scene and the amount of materials and assets used. This is reflected in the results, where the difference in performance between light field rendering and path tracing is lower with more simple scenes. Therefore, light field rendering has a significant potential to be used in 3D simulations.

Fig. 2.4 shows how the results differ visually. The chosen scene contains only diffuse and slightly specular materials so that it can be visually compared. Note that other more complex materials would show more significant changes and the visual comparison focused on the common issues of each method would be more difficult to visualize.

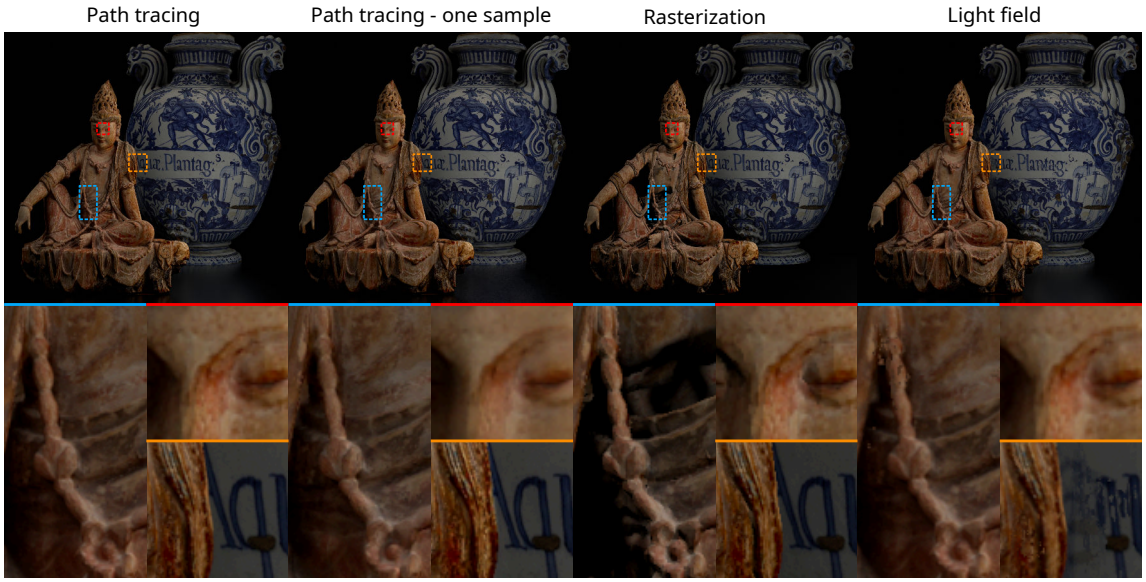


Figure 2.4: A simple scene is rendered using different methods and visually compared in the figure. Problematic parts are highlighted. One-sample path tracing shows blurry results with vanishing details as visible in the red zoom. Rasterization has problems with precise computation of the shadows and lighting effect as visible in the blue zoom. Light field rendering can produce focusing artifacts as visible in the orange zoom.

Fast light field rendering, which is the main topic of this thesis, can be crucial for use cases, such as streaming of light field movies or integration in 3D scenes; see Fig. 2.5.

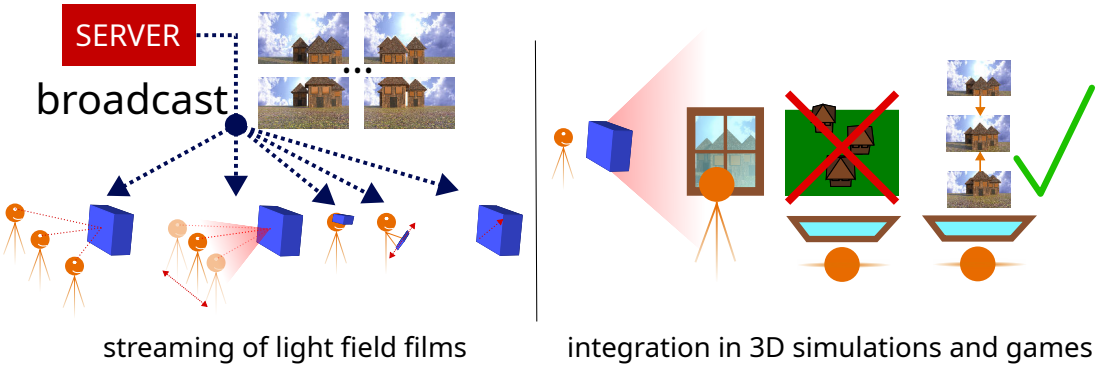
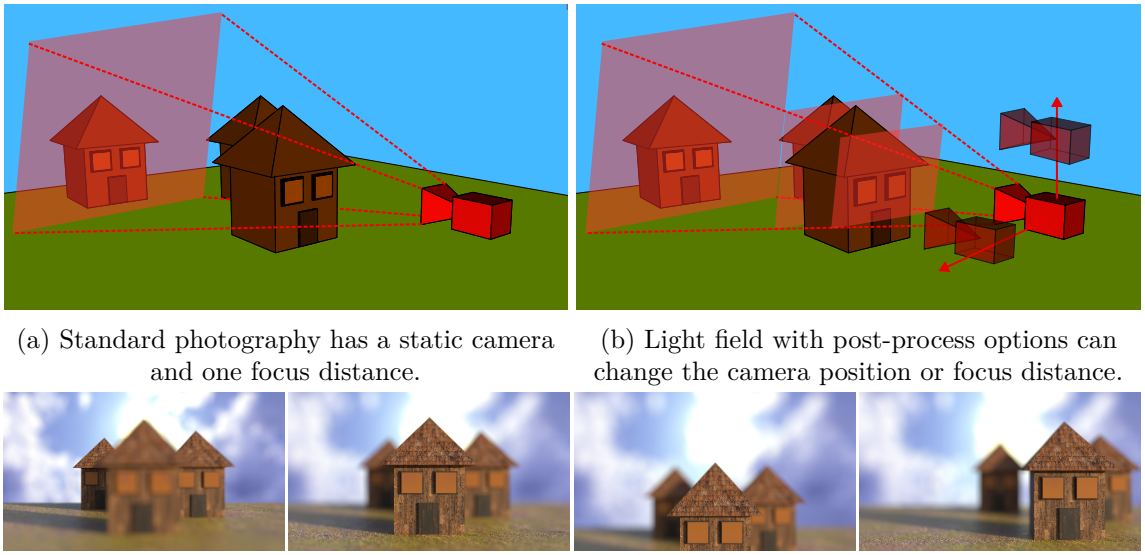


Figure 2.5: Real-time light field rendering, is important for streaming of light field films for various 2D or 3D displays. Any view of the scene has to be accessible by a quick synthesis so the user can change the viewing position. In 3D simulations, light field could replace geometry that is not fully accessible by the camera and would be time-consuming to render.

An example of light field video usage is in cinematography for post-processing purposes. The video, captured as a light field, would allow for a change of focus distance, camera position, and would natively support a stereo or multi-view format export [264]; see Fig. 2.6.



(a) Standard photography has a static camera and one focus distance.

(b) Light field with post-process options can change the camera position or focus distance.

(c) The images are examples of the refocused and differently positioned views.

Figure 2.6: The figure illustrates the difference between the standard photography and light field. Light fields can be refocused and the viewing camera can be moved in the post-processing. The red plane at the geometry depicts the focus distance.

In 3D simulations, like computer games, light field assets would be an optimal way to render computationally demanding scenes [115], where the player's movement is limited to avoid reaching uncaptured parts of the scene [261]. Light fields assets for 3D environment would be optimal for scenes such as views through windows, inside rooms, into boxes with various objects, at distant scenery, etc.

2.1.2 Light Field in Already Existing Software

Google introduced a VR light field showcase application called Welcome to Light Fields [201]. It serves as a virtual tour that shows a real-life captured scene with a rotating camera array. The entire scene is scanned from a particular position. To minimize GPU memory requirements, tiled streaming and caching is used along with modified VP9 compression. A disk-based reconstruction method is used to render the final image. The whole scene is represented by a collection of disk-shaped windows with tessellated geometry based on depth values estimated by multi-view stereo algorithms.

Leia LumePad, a 3D tablet, comes with a LeiaPix application. It is a light field editor that allows unified editing of stereo photos, which are further interpolated into a higher density light field; see Fig. 2.7.

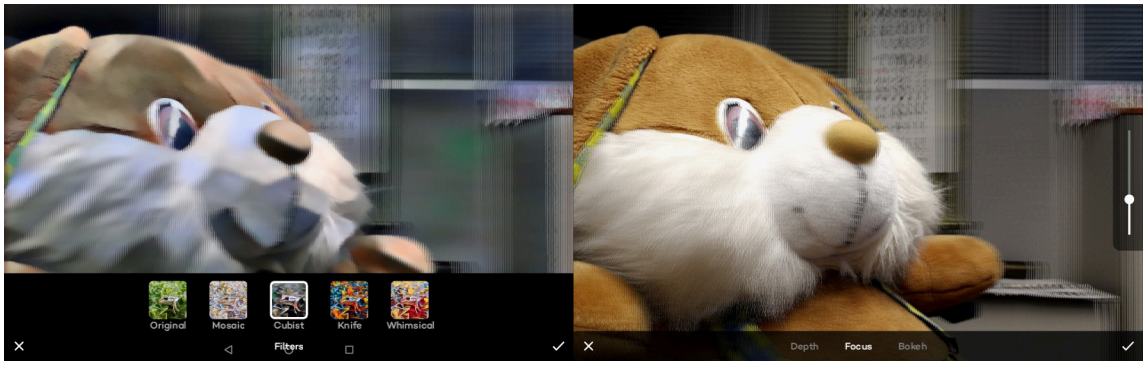
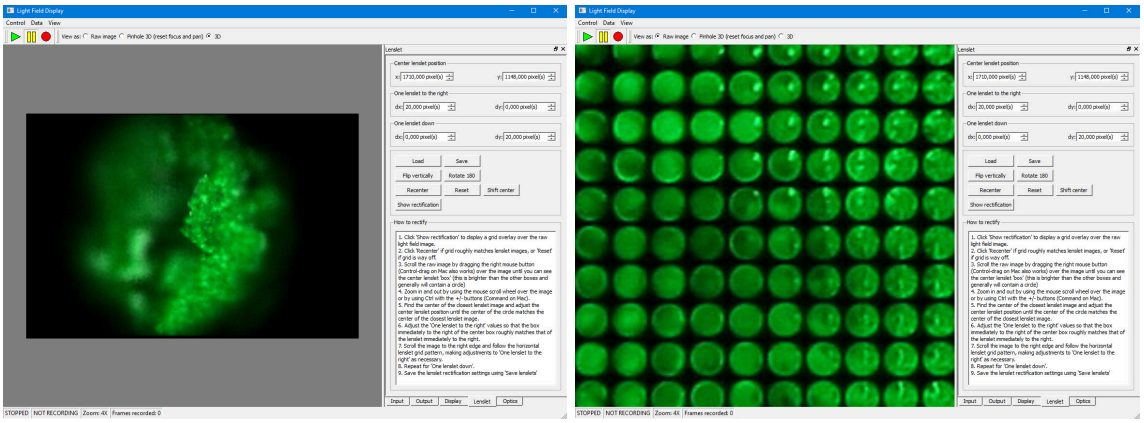


Figure 2.7: The figure contains two screenshots from the LeiaPix light field editing software on 3D tablet LumePad. The images are displayed in 3D and basic edits can be conducted.

An open-source light field imaging application PlenoptiCam [99] exists and allows the user to display photos from Lytro or similar plenoptic cameras. It allows for various operations with plenoptic images such as depth map generation, refocusing or interactive camera motion; see Fig. 2.8. A similar application Light Field Video Viewer [277] also supports viewing light field videos. The author proposed using plenoptic camera along with classic 2D camera to capture the video at the same time and then use the angular information from plenoptic camera to enhance the 2D video adding all the attributes of light field video. In this way, the bandwidth is reduced since all the data from the plenoptic camera are not being transferred and read during the final render. LFDisplay software [159] can be used to view plenoptic images from light field microscopes; see Fig. 2.9. It supports tilting and refocusing of the captured sample. Light Field Toolbox [61] is a set of algorithms used to analyze light fields in Matlab software. It mainly supports Lytro plenoptic images but can load other formats as well. The toolbox can be used to calibrate, rectify, refocus and visualize light fields. Light Field Suite [279] is a set of tools that can be used to estimate disparity from light field views. Similar features for plenoptic camera images are contained in Plenoptic Toolbox [203] that supports rendering of the 3D views from such photos. However, none of these products would be able to replace the proposal of this thesis because of their inability to process streamed, high-resolution, widely spaced light field data in real time.

Figure 2.8: The figure shows import (top left) and result (bottom) windows of PlenoptiCam. The result can be viewed from different viewing angles. Point cloud and mesh can be exported (top right). INRIA plenoptic camera dataset was used [238].



(a) The object can be viewed in 3D. (b) The lenslet light field image can be displayed.

Figure 2.9: The figure shows two viewing modes of microscopic light field images in LFDisplay software.

2.2 Physical Background

In physics, light is defined as electromagnetic radiation of a certain wavelength [65]. Wavelengths $\lambda \in (380, 750)$ in nanometers are regarded as visible light with respect to the human visual system [44, 246]. Light transmission is described by geometrical (ray) or physical (wave) optics [94]. Light is propagated in space by elementary particles called photons. Light can also be viewed as a field defined by a vector function [76, 88]. Specific wave or particle properties of light are not considered in this model. The model describes a field of rays in any possible directions. To model such a field, a 7D plenoptic function [32] L can be defined as in Eq. 2.1.

$$I = \mathrm{L}(p_x, p_y, p_z, \theta, \varphi, m, \lambda). \tag{2.1}$$

This function returns the light intensity I of an incoming ray to a point in 3D space with coordinates p_x, p_y, p_z from a given direction represented by two spherical angles θ, φ; see Fig. 2.10. Assuming a non-static scene, the light can vary for each moment m in time and for different wavelengths λ.

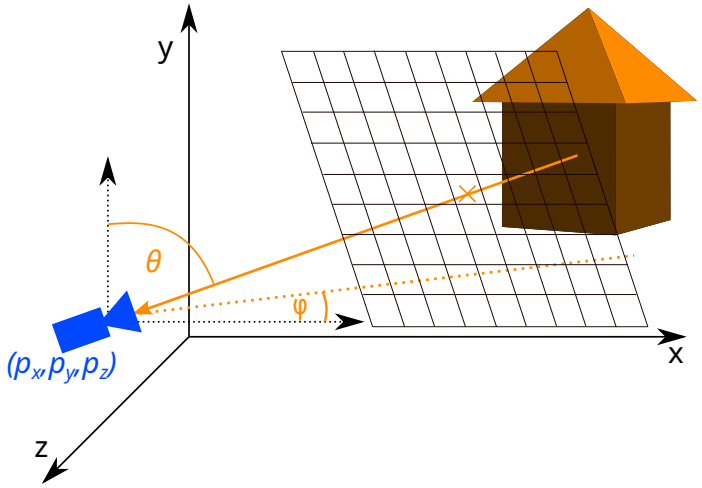


Figure 2.10: The camera position and direction can be parametrized in 3D space as arguments of a plenoptic function. The function can be discretized and sampled by grid.

In physics, the intensity of the light along a ray would be called radiance. The plenoptic function is defined for a point and differs from the radiance, which is defined per unit area. It resembles the definition of radiant intensity. Light fields are also sometimes called radiance fields. The incoming light to the given point can originate from many different sources. The light sources illuminate the scene and the light is absorbed, reflected, or refracted. The intensity of the light of a given wavelength changes due to its interaction with the environment. The resulting intensity can be a result of a combination of several light sources.

The field contains the continuous information about the light so a full reconstruction is always possible when each query point is contained in the light field. Light field defined as a vector field of rays is a theoretical concept. The storage of continuous signal information is challenging in computer science, and several approximation techniques have to be used in practice.

2.3 Discrete Approximation

Light information in computer science is often discretely encoded by a matrix of values of a given color model representing an image [117]. A commonly used RGB additive color model defines light as a vector of three channels: red, green, and blue, which corresponds to the biological processing of light in the human eye [176]. This vector defined for a specific coordinate in an image is called the pixel. Classic cameras usually quantize the incoming light to the sensor to one 8-bit value per channel. High Dynamic Range (HDR) devices, which aim to capture real light more accurately, use higher precision incoming light information such as 10-bit, 16-bit, or 32-bit value per channel [213]. The light field can be approximated by a set of images, where each image represents a view on the scene taken with a different camera position. The views are also sometimes called sub-aperture images.

For static scenes and the definition of the return value I as a light color, for example, RGB vector, the 7D plenoptic function can be approximated by a 5D function. The wavelength λ and time moment m arguments are omitted. In order to efficiently represent a scene by light field, the viewing location is limited to the outside of a convex hull of the scene. Light field would ideally capture a free space without occluders. When viewing only the light field scene without additional 3D environment, the viewing and capturing cameras are placed exactly on the scene convex hull boundary, which can be a simple bounding box. Fig. 2.11 shows how a scene can be enclosed and sampled by virtual cameras as light field. The surface of the convex hull of the scene contains the visual information about the enclosed scene for each direction.

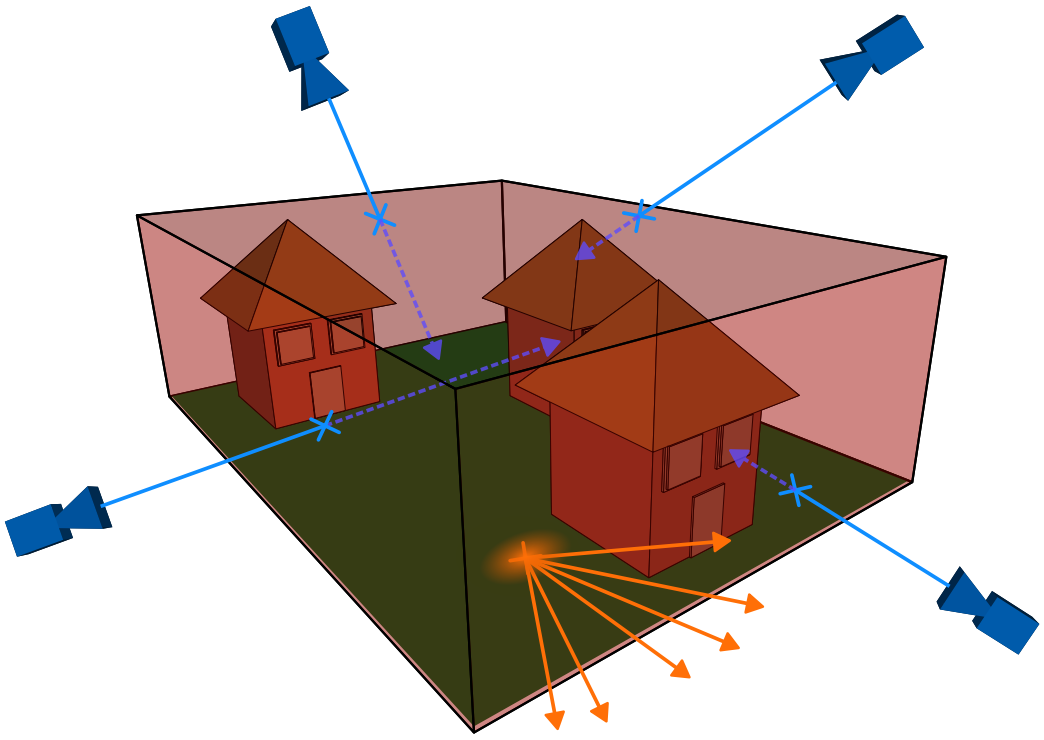


Figure 2.11: The scene is enclosed in a bounding box. The surface of the box contains visual information about the enclosed scene in all directions. The cameras around the box can sample the surface from all positions around it.

When casting a ray from the viewing camera, the light information has to stay constant along the whole ray. In this way, the function can be further simplified to 4D, commonly referred to as the light slab [158] or the lumigraph [90]. The reduced 4D function L' in Eq. 2.2 can be geometrically represented as a ray intersection of two planes or other predefined surfaces [168], for example, sphere-plane [46], direction-point [45], or positional-directional sphere [118].

$$I = \mathrm{L}'(s, t, u, v) \tag{2.2}$$

The ray would intersect the closer plane at a point with local coordinates of (s, t) and the further plane at (u, v); see Fig. 2.12. The (s, t) coordinates, also called angular, on the camera plane are used to identify which of the input views are relevant for the interpolation of the novel view. The views are mapped on the plane since their capturing cameras were distributed over a planar grid. The (u, v) coordinates, also called spatial, on the focal plane are used to sample the correct pixels from the selected views.

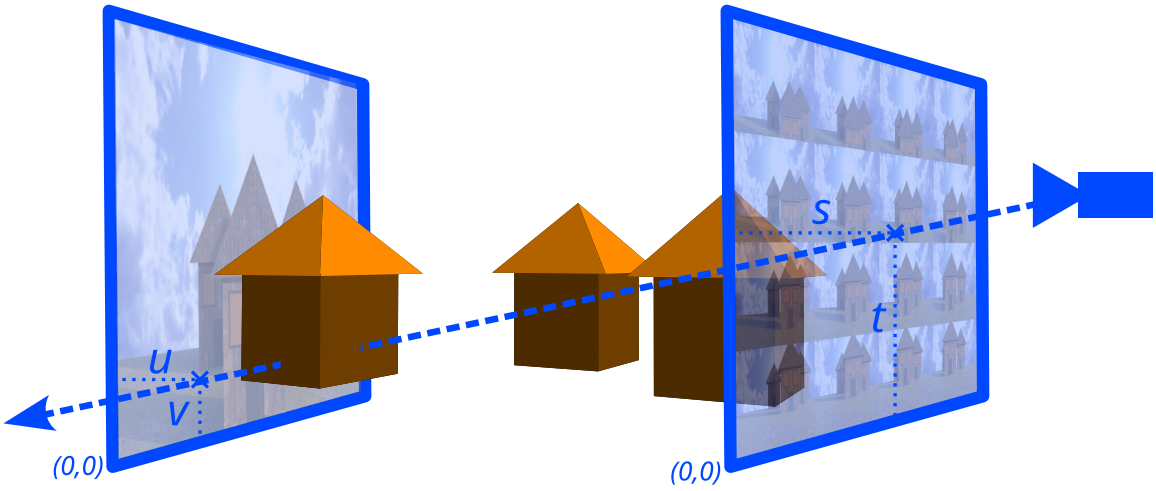


Figure 2.12: The scene is placed between two invisible planes. The rays emitted from the camera intersect the planes to find out which views and which pixels will be used in the novel view synthesis.

The described representations are still continuous in theory. Their dimensions are lower than those of the all-defined light field. All possible orientations of the rays in the described representations cannot be fully defined without errors due to limited memory. That is the reason why only some rays are defined, and the rest needs to be interpolated. In practice, the definition of the rays is implemented by the discrete images. Fig. 2.13 describes the issue of missing information caused by the light field discretization.

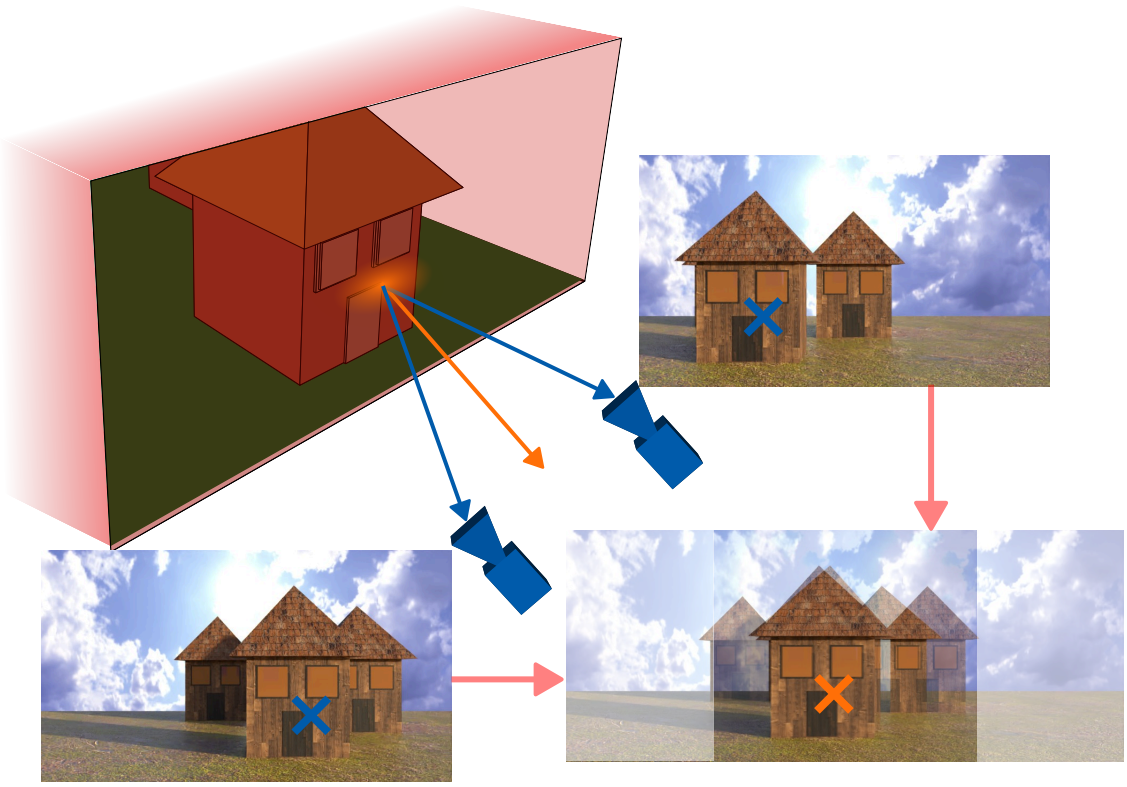


Figure 2.13: Two cameras were used to capture light field in a form of two images. They cover some rays that would sample the light field scene enclosed in a convex hull. The rays that are not directly captured by the cameras need to be interpolated. The pixels from the images can be, for example, simply blended together. The images were shifted so that the highlighted pixels overlap each other.

2.4 Focusing Methods

The light field views can be placed on top of each other and blended together. In such a case, only certain areas of the scene would be clearly visible because the areas that are further or closer to the camera would not be properly aligned in the views. The input images are expected to be without blurred areas, captured with infinite depth of field. The focusing for one pixel in the resulting novel view is defined as a change of the coordinates of the sampled pixels that are used for the final color computation. In this thesis, the pixel in the novel synthetic view is called *focused* if it is interpolated from the pixels, sampled from the input views, that belong to the same 3D spot in the scene.

The resulting view visually resembles a depth-of-field effect where certain parts of the scene are sharp and the rest are blurry. The view can be focused at one distance, or depth information can be used to correctly focus each part of the image. The position of the focal plane determines the focus distance in the scene, that is, which part of the scene is visually sharp; see Fig. 2.14.

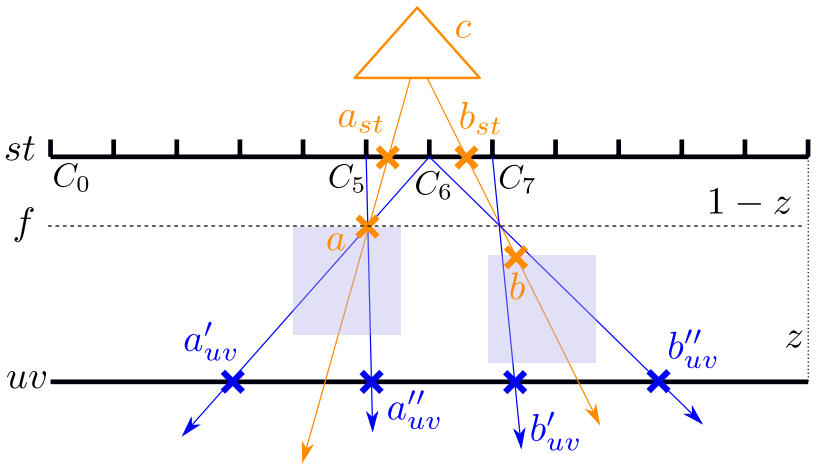


Figure 2.14: The scheme shows side view of a scene with a depth of z. Two rays from the virtual camera c intersect the geometry at points a and b. The capturing cameras c_i are evenly distributed along the st plane. Sampling rays are emitted from the closest cameras (c_5, c_6 and c_7) to the direction of a and b converging at the focus distance f on the camera c ray vector. a_{st} and b_{st} mark the ideal intersection of the virtual camera with the st plane. The intersection points a'_{uv}, a''_{uv} and b'_{uv}, b''_{uv} determine which pixels, from the projected image on the uv plane, are taken into the interpolation. Points a'_{uv}, a''_{uv} demonstrate a situation where the rays intersect the geometry in a correct place. Points b'_{uv}, b''_{uv} intersect the uv plane, ignoring the geometry of the scene that leads to a defocused image.

The change of the focus in the novel view is called *refocusing*. Refocusing is simulated by the interpolation algorithm, which blends the input images together. The nature of the light field refocusing is different from the optical refocusing used in cameras, although it looks similar. In standard photography, light rays that pass through the lens hit the sensor. The adjustment of the lens and the size of the aperture affect how the rays propagate further to the camera body. If the rays converging at the given point on the sensor do not come from the same spot in the scene, such area is blurry. In light field, such an effect is also caused by the color information not coming from the same spot in the scene. The incoming rays on the sensor are simulated by sampling pixels at different coordinates in

the input images. The areas that are out of focus can contain sharp ghosting artifacts due to a low angular resolution of the light field data [49]; see Fig. 2.15. The defocused areas are caused by the disparity of views. A smaller distance between the capturing cameras can reduce the ghosting, but it can also reduce the resulting 3D effect and viewing range. Denser angular sampling can reduce the artifacts, leading to smooth defocus, but increases the space requirements for the light field data due to increasing redundancy between the views.

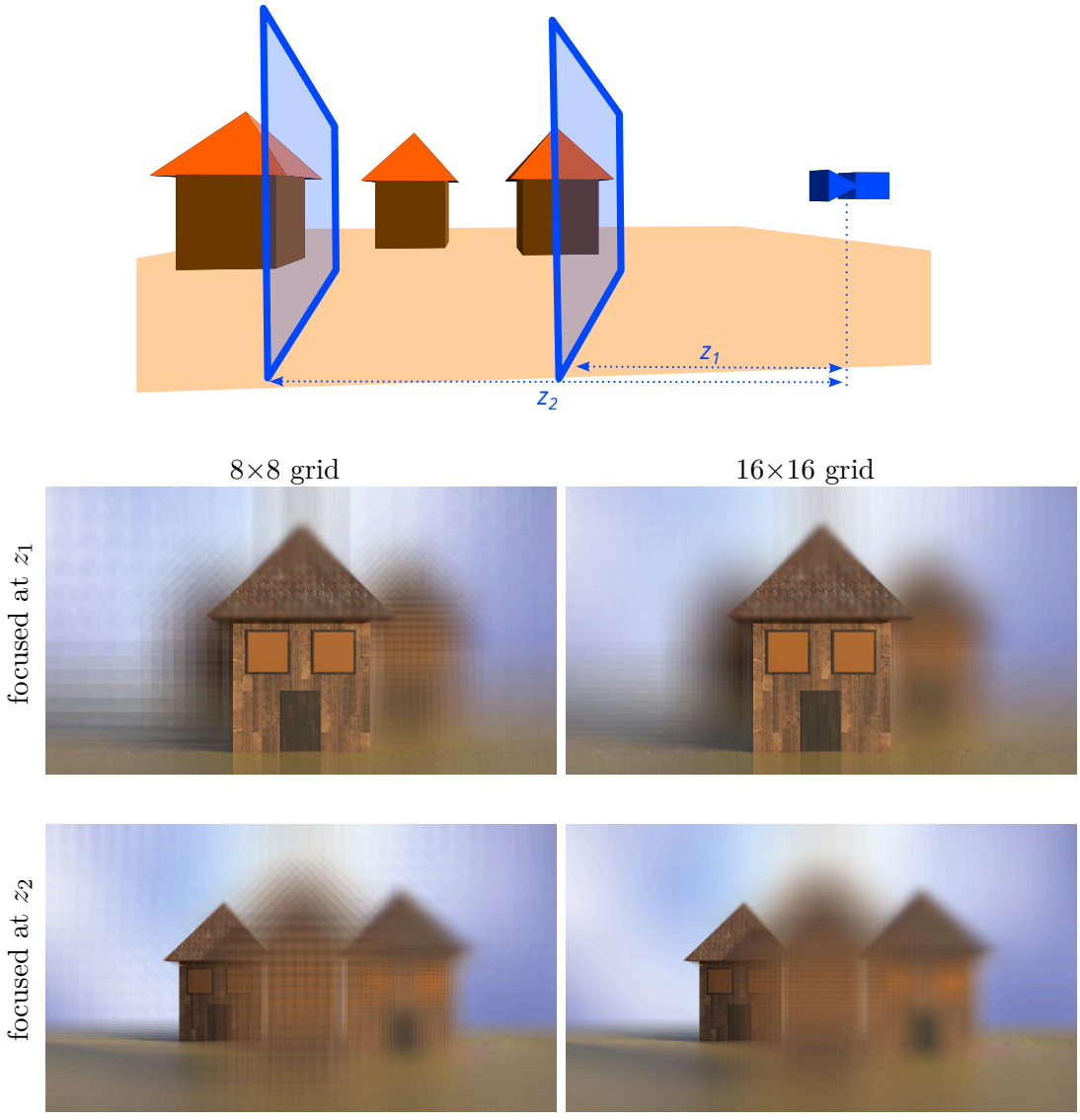


Figure 2.15: The figure shows light field scene focused at two different distances. The amount of artifacts decreases with increasing angular resolution.

In practice, the novel view at angular coordinates s', t' can be produced as a linear combination of shifted input views, weighted by the defined aperture function $\mathrm{A}(s, t)$, using the shift-sum algorithm [13] according to a general Eq. 2.3. The size of the input data grid in the number of images is $S \times T$.

$$\text{view}(s', t') = \int_1^S \int_1^T \text{L}'(s, t, u, v)\text{A}(s, t)\text{d}s\text{d}t \tag{2.3}$$

Compared to the general 4D light field representation, shift-sum works on the basis of pure view interpolation. The synthetic novel view in 4D representation can be produced so that each part of the view can be interpolated from different input views. The intersection of the two planes can inherently identify the angular and spatial coordinates used to sample the input views for each pixel in the novel view. In this way, the whole light field grid can be utilized. Shift-sum uses only the nearest input views to the virtual camera position and is thus less demanding regarding the sampling of the light field grid. This fact can be advantageous for the GPU rendering due to the aligned memory access and SIMD (Single Instruction, Multiple Data) nature where each thread in a warp executes the same instruction on different data.

The refocusing is implemented by changing the sampling coordinates. The sampling spatial coordinates on the focal plane are transformed according to the defined focus distance f. The vector of angular coordinates of the currently sampled view $\mathbf{st}$ and the vector of angular coordinates of the novel view $\mathbf{st}_v$ are used to obtain a new focused coordinates vector $\mathbf{uv}_f$ [195] with spatial coordinates $\mathbf{uv}$; see Eq. 2.4. Note that the signs might change according to the orientation of the view grid.

$$\mathbf{uv}_f = \mathbf{uv} - (\mathbf{st} - \mathbf{st}_v)f \tag{2.4}$$

Eq. 2.5 produces a refocused view_f with the L'_f function extended with the focusing parameter f, internally sampling the input views according to Eq. 2.4.

$$\text{view}_f(s', t', f) = \int_1^S \int_1^T \text{L'}_f(s, t, u, v, f)\text{A}(s, t)\text{d}s\text{d}t \tag{2.5}$$

Eq. 2.3 can be approximated by the weighted sum in Eq. 2.6. The weights w_i approximate the shape of the aperture. They depend on the distance between the sampled view coordinates and the novel view coordinates. They can, for example, be sampled from a Gaussian function with its maximal value at the novel view.

$$\text{view}(s', t', f) = \frac{\sum_{s=1}^{S}\sum_{t=1}^{T}\text{L'}_f(s, t, u, v, f) \cdot w_{st}}{\sum_{s=1}^{S}\sum_{t=1}^{T} w_{st}} \tag{2.6}$$

Fig. 2.16 shows the principle of shifting and blending the images to focus on a certain area in the scene. The focusing process can be performed pixel-wise. Each pixel of the scene can be focused by its own local focal plane; see Fig. 2.17.

If each pixel in the novel view is focused, such view is called *all-focused*. The correct focus distance can be obtained by the ray intersection with the geometry of the scene. A precalculated depth map provides the depth value for each pixel. This value can be used directly or recalculated, according to the capturing camera attributes, as the focus distance [122]. A simplified geometric proxy can be used for the surface estimation instead of the depth maps. The depth can be stored for each pixel in any light field parametrization [262]. This approach uses the principles of ray tracing with depth search [269]. The mutual orientation between the intersection point on the geometric proxy and the arbitrarily positioned input camera can be used without a regular grid to determine which pixels need to be sampled [43].

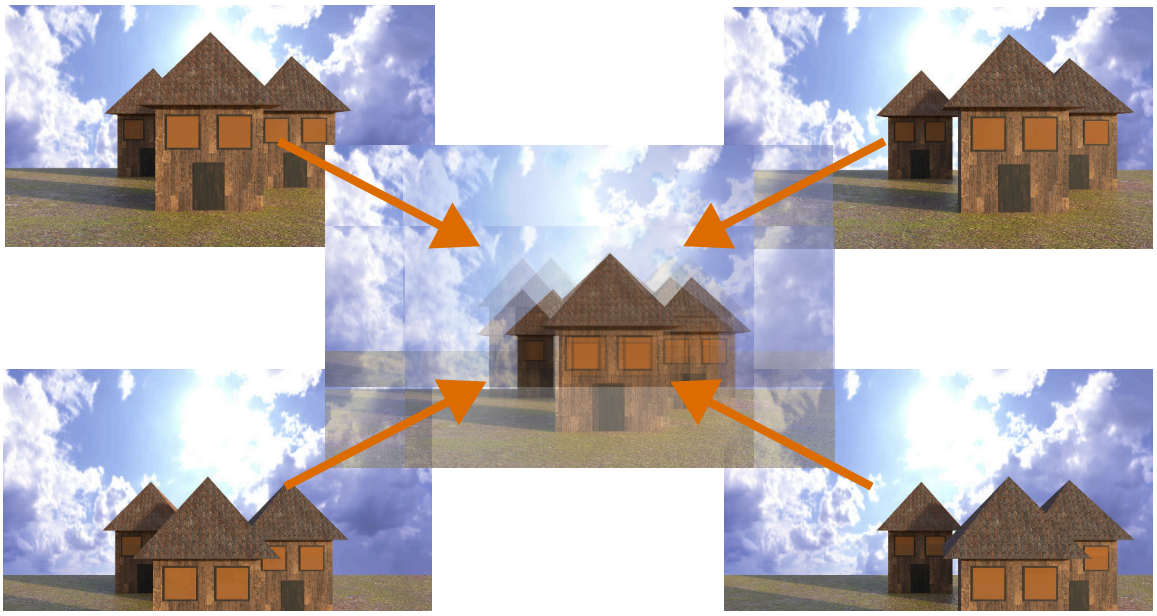


Figure 2.16: Four input images from the grid are shifted to the novel view in the middle of them. They are blended together. The images are shifted so that the roof of the front house is focused.

(a) The scene is focused at the front house. (b) The scene is all-focused, sharp everywhere.

Figure 2.17: The figure contains two views on a 3D scene, demonstrating the effect of all-focused scene. The first one is focused on the front building with a smooth depth-of-field simulation. The second one contains all parts of the scene in focus. Classic 3D rendering methods automatically produce all-focused results and focusing needs to be simulated with additional methods. Simple light field rendering supports only one focus distance and all-focused results need to be produced by more advanced methods.

2.4.1 Depth Extraction for Light Field Focusing

Depth maps can be available during the capture process. Their extraction is straightforward with synthetic scenes, where the scene geometry is known. Depth is usually the position of a pixel in a 3D scene on the axis perpendicular to the camera grid plane. Sometimes, values in depth maps are distances of the pixels from the center of camera projection. Fig. 2.18 shows how the input light field data with depth maps would look like.

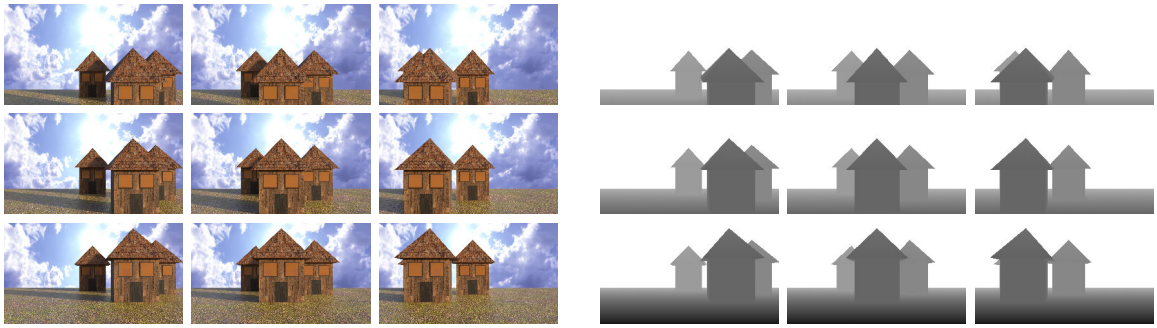


Figure 2.18: The depth maps (on the right) can be used to approximate the scene geometry in light field rendering.

Depth, disparity, or optical flow estimations are dual problems in this case. Each one of them can be used to calculate the other with additional information, such as camera positions and calibration data. The relationship of features between the images serves as a basis for these estimations [31, 121]. Disparity maps describe how much pixels move between two images. This is similar to optical flow that defines vectors for pixels or blocks that describe their relative motion between two similar images.

When optical flow algorithms are applied to multiple camera outputs instead of one camera video, it is possible to calculate a scene flow [183] that partially describes the geometry. It can be said that the scene flow is a combination of disparity/depth and optical flow [233]. The optical flow has to be filtered and the outliers are removed by energy minimization methods [252]. Optical flow calculation is a search problem in which a corresponding block of pixels needs to be found for each block of the other image [114]. Optical flow can be estimated from light field images in a manner similar to stereo views processing [54]. The optical flow calculation is optimized by first performing a sparse matching and then so called spatio-aware edge-aware filter is applied estimating dense flow. The resulting disparity maps are then aggregated using median filtering followed by one-step of energy minimization calculation. Fig. 2.19 demonstrates the visual relation between two views of a light field grid.

Deep learning can also be used to estimate the depth from a set of images or even a single image [70, 207], but additional processing would be needed to make the depth maps consistent throughout the whole light field according to the 3D scene. The principle of light field refocusing can be used to segment the image according to the resulting foused areas to estimate depth [300].

A semi-global matching method estimates a dense disparity from rectified stereo images by searching for most similar pixel blocks between the images in predefined directions and search range [112]. Fast and good quality depth maps generation for light field images using semi-global matching principle was previously proposed [18]. It uses principles of

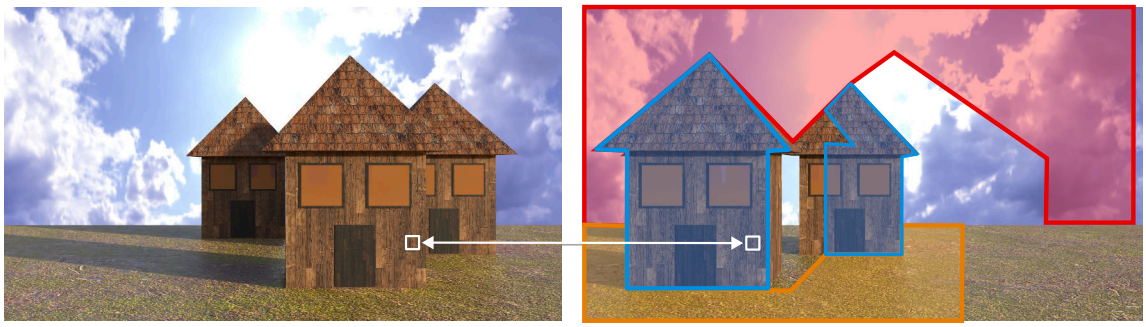


Figure 2.19: The right image contains highlighted areas which are also present in the left image. This redundancy, typical for light field, can be exploited to estimate the depth of the scene. Block-based approaches can be used to find the same spots in the views as depicted by the white drawing connecting both images. The redundancy, on the other hand, increases the amount of data.

the disparity map approximation method for stereo images. This method searches for best disparity values along multiple corresponding lines in the input images.

A lightweight approach [128] manages to extract depth maps even from sparse input data using intermediate disparity images and achieves the same results as more complex, previously proposed, methods in less time. This method uses four corner images from the grid as input. These images contain most of the geometric information about the scene because they are captured from the most extreme positions. Rough disparity maps are estimated from them using optical flow and are aggregated by energy minimization algorithm. After applying bilateral filtering, the maps are warped into the rest of the views. Possible holes that appeared due to occlusions are filled by global inpainting. However, the missing information can produce unwanted visual artifacts in possible view interpolations. The maps can be directly used to reconstruct a 3D geometry of the scene for the given point of view; see Fig. 2.20.

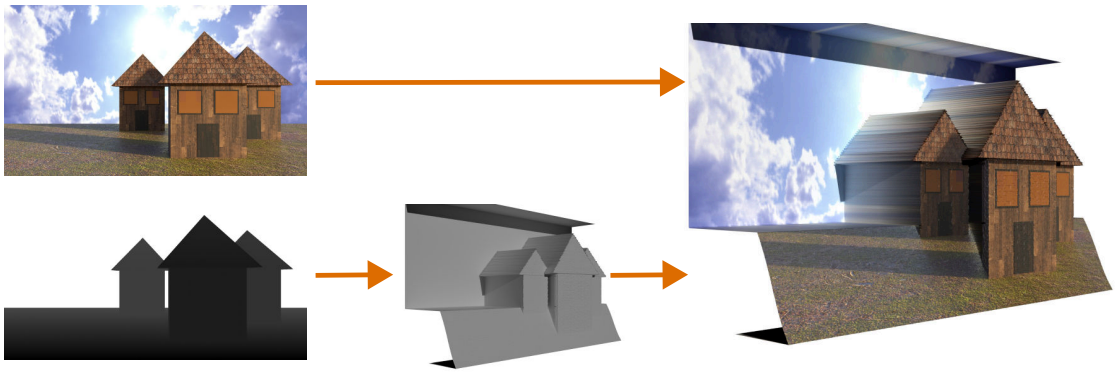


Figure 2.20: 3D mesh can be constructed from depth map but such a mesh is limited to the same or similar point of views as in the depth map image.

Graph Cuts method for the energy minimization for multi-camera scene reconstruction exists [143]. Plenoptic camera datasets can be processed to extract the depth with symmetry property of the focal stack [167]. Focal stack representation of light field data consists of images that are focused at different focal distances with depth-of-field effect. Another

technique suitable for plenoptic camera data uses spatial variance after angular integration of the epipolar image for defocusing and angular variance for correspondence depth cues estimation [257]. Small matching windows can be used for a lot of images in the light field datasets. Flat uniform regions which are not suitable for such approach can be analyzed in a lower resolution, leading to multi-resolution matching approaches [194]. The extremely narrow baseline in lenslet light field camera data causes problems when estimating depth or disparity. This problem can be solved by exploiting phase-shift theorem in the Fourier domain to estimate sub-pixel shifts [124]. The performance of depth or disparity estimation methods is, in most cases, not sufficient for real-time usage along with rendering. Optimized methods for light field data also usually work well with the plenoptic camera data but not with the large baseline datasets. Deep learning methods also require excessive memory.

The multi-resolution depth estimation can also be used when working with wide-baseline sparse datasets. The capturing and rendering pipeline using such an approach with a point-cloud projection-based final image synthesis has already been proposed [223]. Point cloud rendering also requires additional refinements such as smoothing of edges and filling of occluded or empty areas [221]. However, using more cameras than the 4×4 proposed grid to achieve better rendering results in this pipeline could negatively affect the performance. The conversion of depth maps to point cloud is straightforward; see Fig. 2.21.

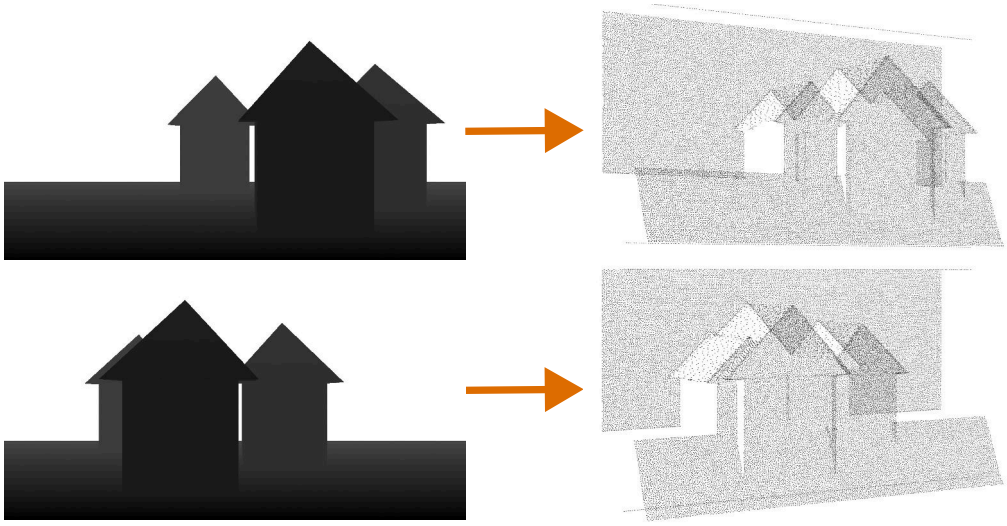


Figure 2.21: The light field depth maps can be converted to point clouds which then can be merged to a 3D scene.

If the depth of the scene is known, novel views can also be rendered using image warping [232]. Light field rendering is potentially more GPU friendly and can maintain constant speed. Warping techniques must solve issues with regard to the ordering of pixels in occluded parts and the search for pixels in areas with missing data during the reprojection [155].

All search problems in algorithms described above can be optimized using the rules of epipolar geometry [289] which can create search bounds to find the corresponding pixels. For such an approach, the camera parameters and their exact positions are needed. One point on one image can be found on a line on the other image.

2.4.2 Light Field Focusing without Input Depth

Additional depth maps can improve the quality of light field rendering. However, they increase the memory and streaming bandwidth requirements and also the number of texture read operations, which is a crucial performance parameter for GPU algorithms. Even if depth maps were available for each light field image, the depth information could be missing for certain, previously occluded, parts of the novel view. The memory requirements can be partially solved by selective streaming of the light field data only for the currently visible part of the scene [74]. Light field rendering methods that do not rely on the input depth maps might work as alternative solutions.

A simplified geometry of the scene can be used in a classic rendering pipeline with the fine details substituted by a texture. To simulate view-dependent material attributes and to mask geometry inaccuracies, each polygon of the scene can be rendered with variable texture [62]. The method determines for each polygon in which input views it appears by projecting them on image plane from the corresponding viewing angle. Partially visible polygons are split. Polygons that are not visible in any of the input images are filled with interpolated colors from neighboring polygons in object space. For each polygon, a view map is generated simulating viewing hemisphere assigning closest input images to the viewing angles. This method combines both image-based and geometric rasterization rendering. Fig. 2.22 demonstrates the principle of variable texturing.

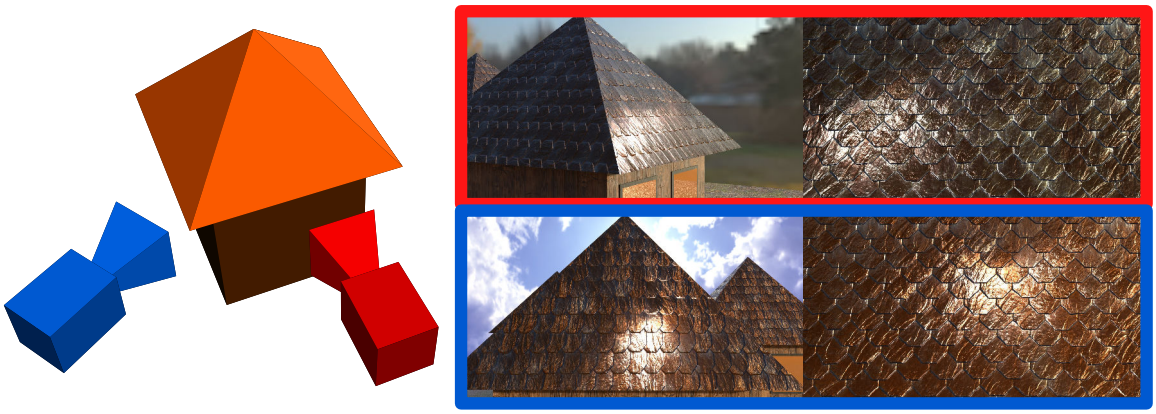


Figure 2.22: A simple geometry can be rendered with dynamic textures that change according to the viewing angle. It is a combination of image-based rendering and rasterization. The images on the right correspond to the cameras and show the captured view (left image) and used textures for the same mesh area on the roof (right image).

Multiple unsynchronized video streams can be used as input for light field rendering [272]. In the first phase, a sparse, bidirectional optical flow is used to find correspondences between the input videos. The result can be refined by applying an epipolar constraint that can serve to detect outliers. Time offsets can be calculated to synchronize the video streams, and temporal interpolation of new video frames is carried out using the image morphing method. The quality is improved by an additional calculation of virtual edges between corresponding feature points. For spatial interpolation, the unstructured lumigraph rendering [43] method is used. This method requires a geometric proxy of the scene, which requires preprocessing of the data with 3D reconstruction or depth extraction algorithms.

A light field movie can first be rendered offline using only the necessary set of cameras to cover the entire scene of interest [145]. Each camera acquires a color and depth cubemap, which is compressed using a similar concept to video compression methods. Raymarching algorithm is used to render the scene in real time using only necessary parts of cubemaps from the original cameras. Textures from cubemaps are chosen according to the distance of the original cameras from the current virtual camera, viewing angle, and field of view.

Tens of differently focused views for a given viewpoint, using standard light field rendering methods can be analyzed to produce an all-focused view [254]. Areas in focus are then chosen [247] from the previously generated views and the final image is constructed from them; see Fig. 2.23. However, this approach was demonstrated only on small-resolution images with a small distance between the cameras. It also uses multiple synthesis filters, exploiting the density of Lytro dataset, which might not work well on the sparse datasets. The method was further improved, but it is still unusable for real-time rendering [148]. This principle is similar to the main proposal of this thesis. However, the proposed method works even with sparse data and on a per-pixel basis suitable for GPU acceleration.

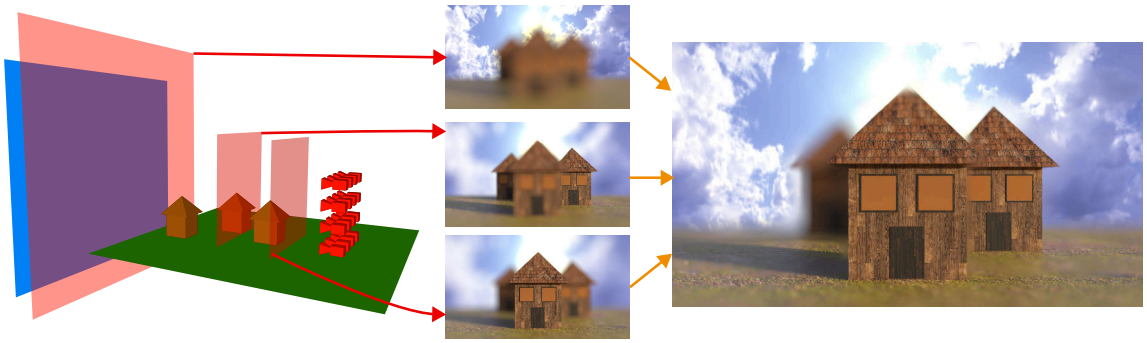


Figure 2.23: One possible method to achieve all-focused novel view is to produce several views that are focused at different distances. Then composing the final view from the focused parts of each intermediate view.

All-focus image can also be generated using high dynamic range light field [160], where the position, direction, and exposure time information is integrated in the light field model. Local focus distances can also be estimated for each view or even for each triangle of the resulting viewing plane by minimizing the least squares error [107].

In a more general way, the principle of selecting the optimal focus distance per pixel or pixel block can be defined by the 3D cost volume model [93]; see Fig. 2.24. The volume is defined by its resolution, which could be the same as the desired pixel resolution of the novel view. In general, the resolution is the density of the result and higher resolution provides higher quality results but is more computationally demanding. The depth of the volume is the hypothesis range. The hypothesis planes are possible results that are being searched. The search algorithm uses a cost function to determine which hypothesis is the optimal one for the given spatial coordinate. Note that this model is ideal for massive parallelism computations, such as with GPU architectures. This model is used for disparity or depth estimation in multi-view processing, as well as in the main proposal of this thesis.

Light field images can also be analysed in spectral domain. One of the depth-independent reconstruction methods exploits the sparsity in continuous Fourier domain to sample the light field efficiently [239]. A new sampling quality metric that outperforms the

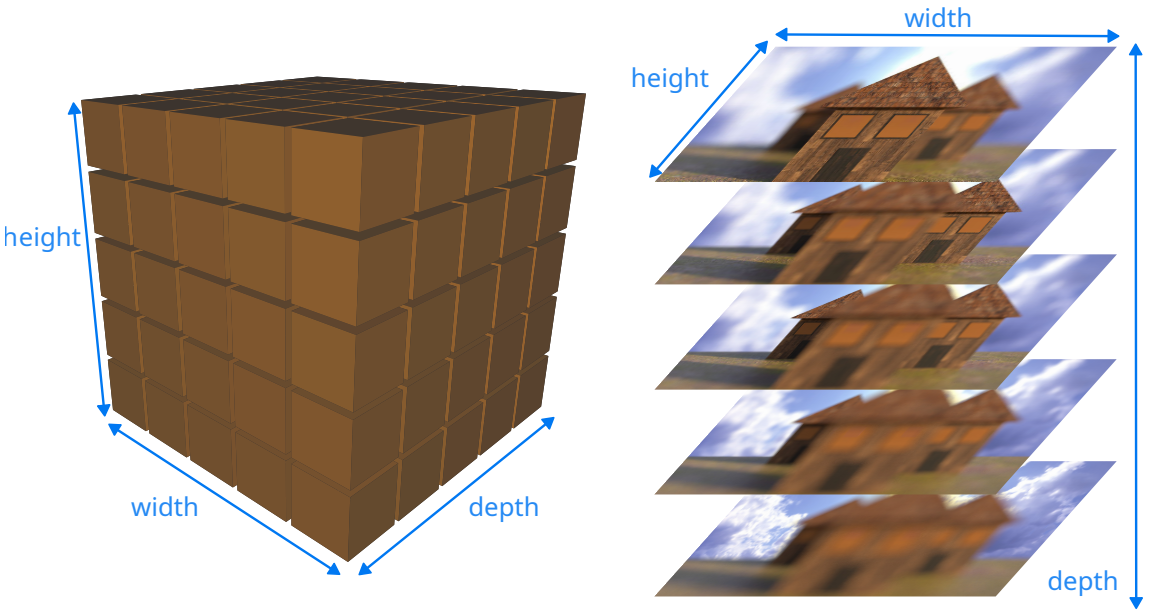


Figure 2.24: 3D cost volume consists of cells that can be mapped to pixel blocks of light field focal stack. Each hypothesis plane in the depth dimension could be a differently focused light field view. The goal of a search algorithm would be to identify the optimal hypothesis for each spatial coordinate.

maximized minimum distance and reduces the search space, using symmetry constraints, can be used in the novel view synthesis [231].

Densely sampled epipolar-plane image (EPI) reconstruction using shearlet transform can be achieved exploiting the light field sparsity in shearlet domain [266]. EPI is a visual representation of the scene where a visible point appears as a line with slope derived from the distance between the capturing camera and the point itself. The intensity along the line defines the light emission from the given point. The depth of the scene is visible from the EPI [156] as shown in Fig. 2.25. This representation is the key principle of the EPI reconstruction method. Depth-guided filters are used to analyze the EPI in the frequency domain to identify the optimal depth layer that can be used for the novel view reconstruction.

Deep-learning rendering based on few reference images [75, 104] can be used to produce novel views from light field data. An unsupervised approach working with planar light fields, using one network for disparity and one for occlusion map estimation, managed to yield results comparable to supervised approaches, overcoming the drawbacks of the full supervision methods [196]. The disparity estimation uses pyramid and a cost volume approach on the input images. Warping is used to produce a novel view. The warping is used in a cyclic manner to acquire the information about the scene geometry and occlusions. Forward-backward warping takes two input views, warps one to the novel view, from the novel view to the second input view, and then back to the novel and first input. In this way, inconsistencies between the warped pixels and the input views reveal possible missing information caused by occlusions; see Fig. 2.26.

Specific deep learning models for light field data and novel view synthesis also exist [249, 250]. When the camera position and parameters are known, image features can be sampled along epipolar lines across several views near the virtual one. The neural network can evalu-

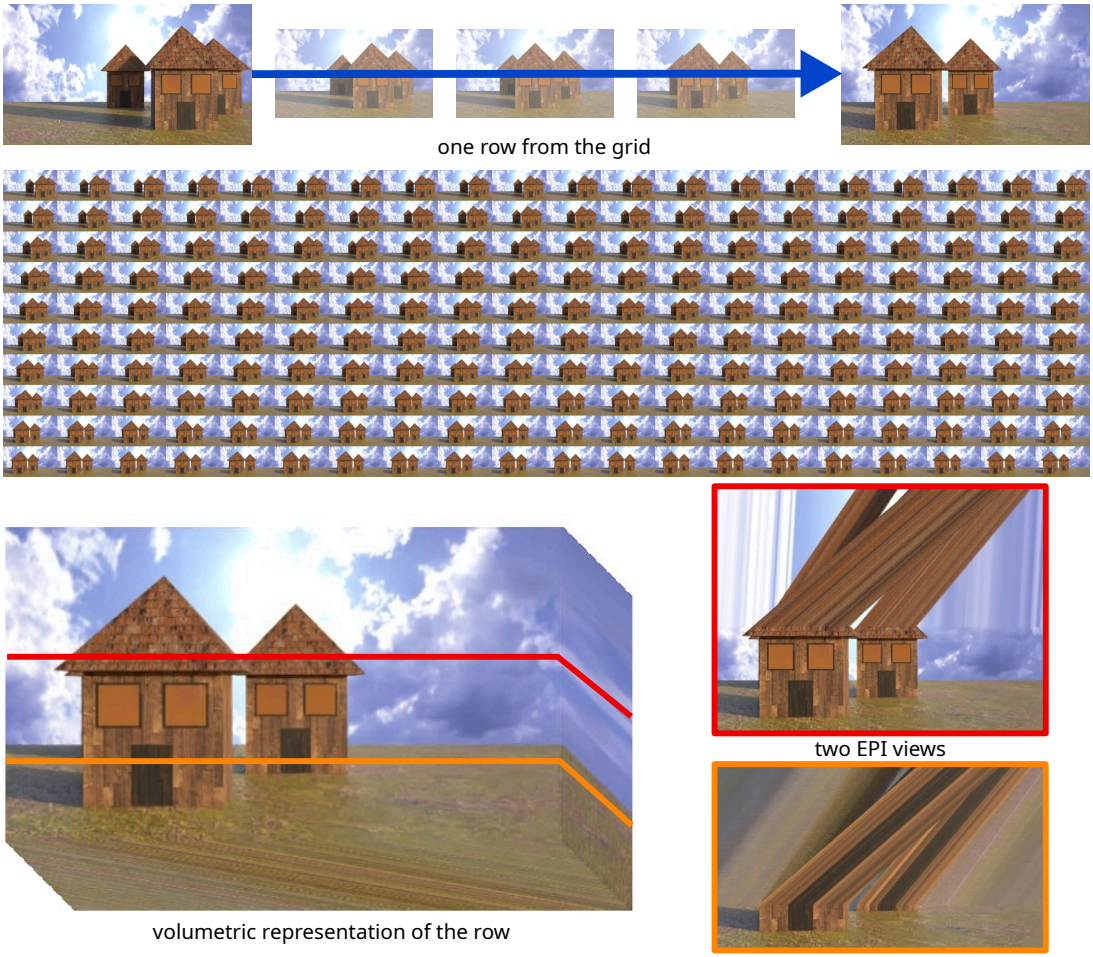


Figure 2.25: Epipolar-plane image is constructed from a dense light field, where one row is selected, views are sorted and stacked on top of each other. The views are cut and displayed as a slice through the volume. The geometry of the scene is visible in the EPI in the slopes of the lines. The intersections of the lines mark occlusions. Two EPI views are shown from two slicing positions.

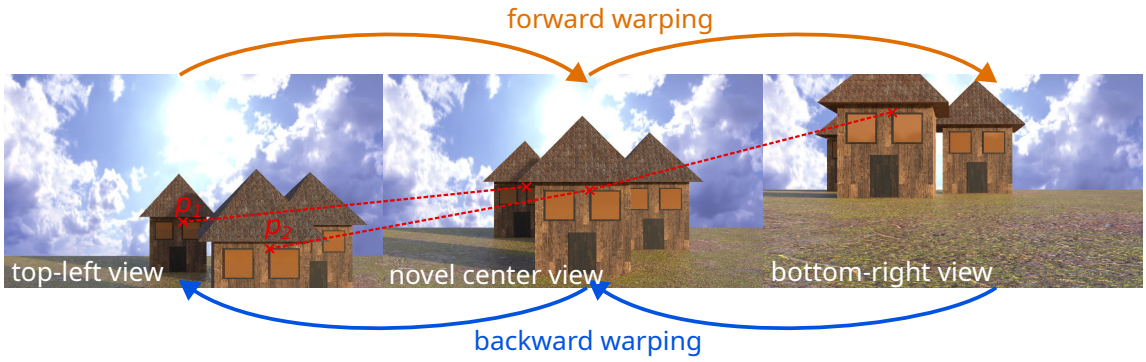


Figure 2.26: The middle image is the novel view produced by blending and warping of the left and right input views. To reveal occlusions, forward and backward warping is used so, for example, point p_1 can be warped from the first input to the novel view but not to the second input. That reveals p_1 being occluded in the second input. Point p_2 is visible in all views and would be propagated through the warping back and forth.

ate the best match along these lines to synthesize all-focused view. These methods can deal with occlusions even with missing data. Their real-time usage in 3D simulations with high-resolution light fields would be problematic due to their memory and time requirements. Fig. 2.27 shows the principle of epipolar features.

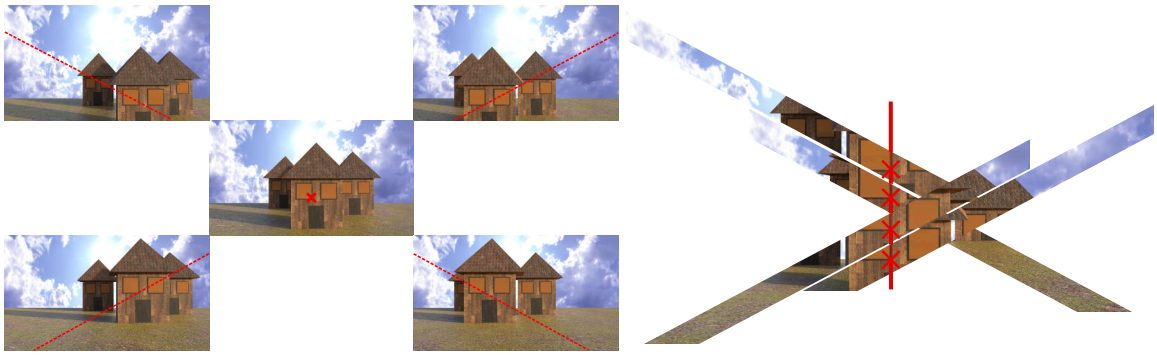


Figure 2.27: A point in the central novel view has related points in the input view images from the same spot in the scene if no occlusions are present. Many light field rendering methods exploit the fact that the positions of the input views are known and an epipolar line can be defined for each image. The main task of the rendering is the search for the best match along epipolar lines (right).

Tremedous amount of research was conducted in the field of video frame interpolation [66]. The frame interpolation methods can be used to enhance a video, especially by increasing its framerate. Such videos look more smooth. Frame interpolation is a common way to improve framerate even in the field of computer games [137]. Less information is needed to be transmitted via the network or from the host memory to the GPU. Also, fewer views need to be rendered in case of 3D simulations with graphical output. Frame interpolation methods can also be used for the novel view synthesis in light fields. Fig. 2.28 shows the approach to exploiting frame interpolation, which usually works as a novel frame interpolator between two consecutive video frames. The interpolation is used three times to acquire a central view between four nearest views in the light field grid.

The standard frame interpolation can be implemented using optical flow, which identifies the corresponding parts of two frames and serves as the interpolation guide. Fast GPU accelerated approaches exist, such as NVIDIA FRUC library[1] which can be directly used to interpolate intermediate video frames. A GPU-accelerated high-accuracy optical flow [41] is implemented in OpenCV framework and can be used to interpolate, for example, new video frames between two original ones. This approach can be used for light field data to make the input image grid more dense or to generate intermediate views. The limitation is the processing time and the inability to use additional information from surrounding views, except for the two reference ones. Utilizing optical flow-based methods can be problematic when the scene contains a lot of occluded parts, reflections, or areas without detailed textures.

Deep learning provides better results in such cases. The frame interpolation seems to work better on light field data than general deep-view interpolation methods [8]. Light field data usually do not capture the scene from significantly different angles and positions, which makes them unsuitable for methods designed to work with free-look data.

[1] developer.nvidia.com/opticalflow-sdk

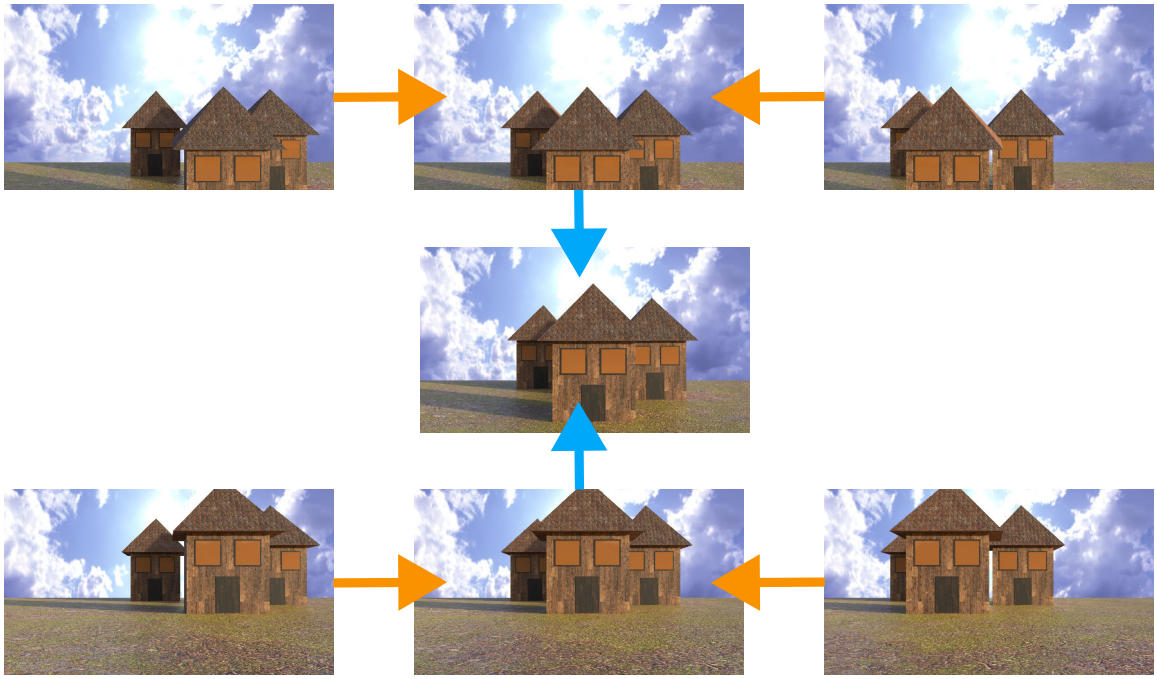


Figure 2.28: The interpolation is first used to synthesize the middle images between two pairs. The results are then interpolated again into the final view in the middle of the four input views.

A combination of meta-learning and test-time adaptation leads to a robust network for frame interpolation [57]. Meta-learning ensures better adaptability for various input tasks as it modifies the learning algorithm itself according to the given task. Frame triplets were used as the training samples. The triplet consists of three consecutive video frames, where the middle one is considered as the expected ground-truth result. Optical flow and pyramidal feature extractors can be combined with deep learning and softmax splatting, which guarantees better interpolation results than optical flow or deep learning alone [197]. Bidirectional optical flow between the two input images is first estimated. The network is used to synthesize the novel view between them, guided by the warped images according to the flow. Softmax splatting solves the problems with sampling and splatting; see Fig. 2.29. It can take into account the depth relation between the pixels and separate background and foreground correctly.

FLAVR framework uses 3D spatio-temporal kernels to directly learn motion properties from unlabeled videos [135]. This method can provide fast results in a good quality. The improved texture synthesis in ST-MFNet outperforms the previous approaches in terms of visual quality and uses four closest frames in the video instead of two [60].

General multi-view synthesis neural approaches can also provide acceptable results [190, 278] but are often not designed for light field data where views are aligned in a grid. To reduce the memory requirements, it is optimal to use only the nearest views, which seems to be problematic for this kind of network. The problem is similar to the 3D reconstruction methods, which need more 3D information about the scene derived from the views. These issues can be partially solved by identifying the camera rays projected in all views. The color information along the rays can be used as the input of the network which significantly shrinks the search domain for the correct synthesis of the novel view [274]. IFRNet is a

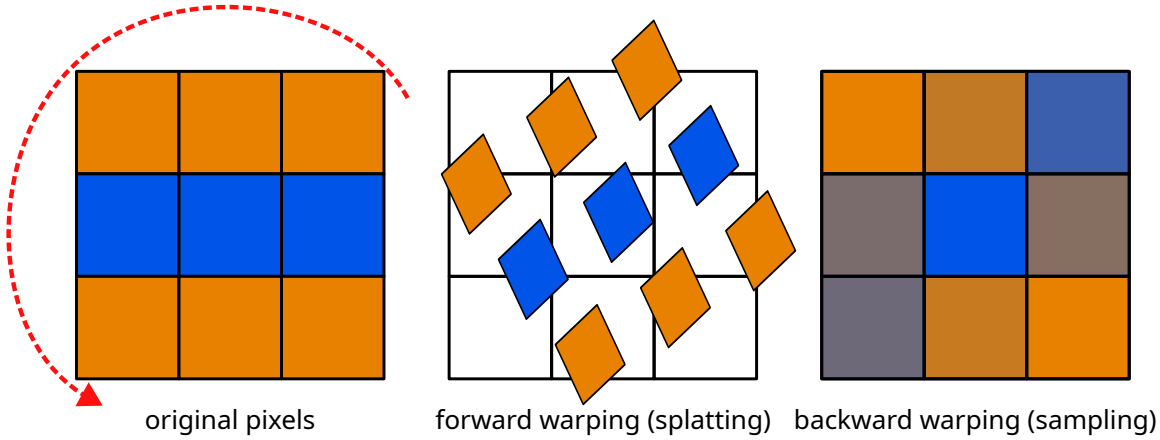


Figure 2.29: When image is transformed, for example, warped by disparity map to the novel camera position, the result can be produced in two simple ways. By splatting, sending the input pixels directly to the resulting pixel grid which can produce holes and overlaps. By sampling, where each pixel in the destination grid looks for the nearest source pixels and computes the new value, for example, by averaging them. The figure shows how rotated pixels are propagated.

real-time solution that reaches the same quality levels as the other works [144]. Similarly to other approaches, feature pyramid is extracted from the input views. The output of the feature extraction encoder is passed to the coarse-to-fine decoders. Intermediate flow fields are gradually refined in a sequence of decoders. The performance of the model is improved by novel objective functions: task-oriented flow distillation loss and feature space geometry consistency loss. Pyramidal approach is used in computer vision often to detect features and objects at different scales; see Fig. 2.30.

The deep interpolation is ready to be used even for modern standards like 4K videos [204]. Large spaces between the frames and high resolution of the video introduce drawbacks to the existing methods. The main issue is larger pixel distance between the features in the frames. Search windows would have to be excessively big to quickly match the detected inter-frame features. Global motion fields at a coarse scale are computed. The issue of vanishing details in coarser scale is solved by refinement of the fields with an upscaling module. Fig. 2.31 shows the problem of uneven distances between original and downscaled images.

Networks such as FILM [216], that support large motion between the frames are the most relevant for light field synthesis, as they are not limited to close views only. Input images are processed in a pyramid manner with a feature extractor. Scale agnostic bi-directional flows are computed from the features, and the final view is synthesized by UNet decoder.

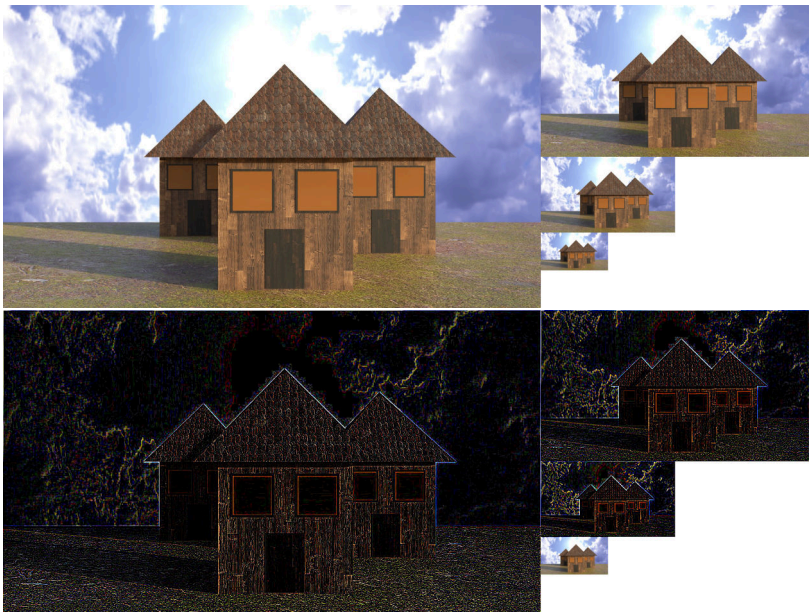


Figure 2.30: The input image is scaled down multiple times to produce several levels of scale in the Gaussian pyramid (top image). This pyramid can be used in scale-invariant feature or object detection. Laplacian pyramid (bottom image) is formed as a subtraction of two successive levels in the Gaussian pyramid and is used for image compression. The pyramids are often produced in grayscale, this figure contains RGB versions and its brightness is increased for better readability.

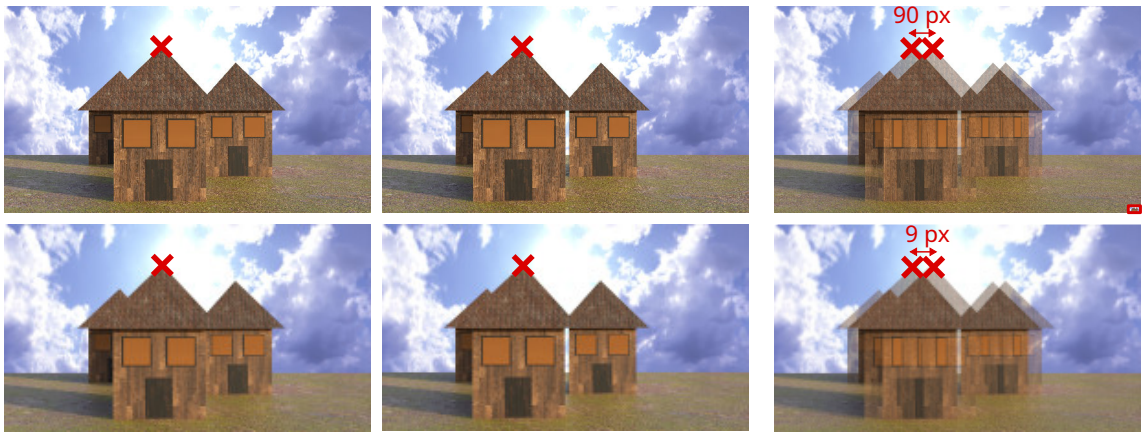
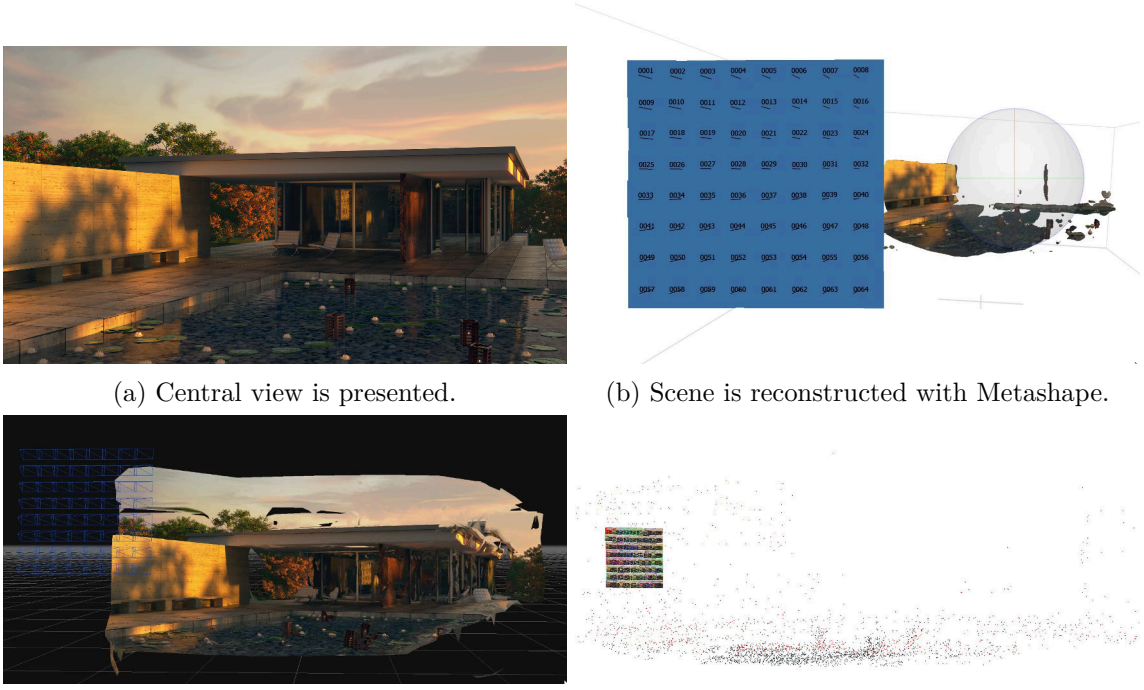


Figure 2.31: Search windows used to find the corresponding features in two views might not be optimal for high-resolution inputs. The image can be downscaled to reduce the distance in pixels between the features. Important details might be lost in such process due to the general loss of information introduced by downscaling. The figure shows the original and 10× downsampled views. Right bottom corner of the original blended image (top-right image) shows the size of their difference.

2.4.3 3D Reconstruction

The light field images can be used in geometry reconstruction algorithms [188]. The quality of the results depends on the density of input views, existence of additional information, for example, depth and camera parameters, lighting conditions, or image quality. Standard photogrammetry and structure-from-motion methods, where the 3D scene or object is reconstructed based on the corresponding features and similarities detected in the input pictures [169], can be used with light field data. Software such as COLMAP, Meshroom, VisualSFM, Regard3D, Metashape, Autodesk ReCap, 3DF Zephyr exists and can generate 3D representations of the light field scene with quality dependent on the dataset and methods implemented in each program. However, according to the conducted experiments, these methods might fail with light field datasets because of the lack of free-look views around the scene. Fig. 2.32 shows the results of 3D reconstruction experiments on standard light field datasets.



(a) Central view is presented.

(b) Scene is reconstructed with Metashape.

(c) Scene is reconstructed with 3DF Zephyr.

(d) Scene is reconstructed with Visual SFM.

Figure 2.32: The existing reconstruction software does not work well with light field data. Metashape estimated the camera positions correctly and produced an rough and incomplete 3D mesh. 3DF Zephyr reconstructed the scene and produced an accurate mesh estimation in the shortest time, but the model contains a lot of holes. Visual SFM estimated the camera positions, reconstructed the sparse point cloud but could not produce a viable dense cloud. The reconstruction took several minutes, measured with Intel® Core™ i9-9900K and NVIDIA® GeForce RTX™ 4090.

COLMAP, Meshroom and Regard3D were not able to produce an acceptable result. Most of the scenes that work well in light field rendering methods are not usable at all in the reconstruction software, even with high-quality settings. The reconstruction sometimes takes tens of minutes to complete, even on a high-end machine. These approaches need to

preprocess the input data, calibrate the camera, find the corresponding features between the images, create the mesh, and apply textures [136]. Generally, light field datasets with wider space between the capture cameras are more suitable for 3D reconstruction algorithms. Light field rendering methods, on the other hand, generate more artifacts with such datasets.

It is possible to reconstruct a scene in real time, for example, with a stereo camera [85], but the possible combinations of light field images increase the necessary processing time. Real-time methods such as ORB-SLAM3 [47] for camera pose estimation can usually also produce a dense point cloud of the environment that can be used for 3D reconstruction. These methods usually depend on the already calibrated camera parameters, and the process should be initialized with enough input views of various orientations [256]. A similar approach to 3D reconstruction is point-cloud reprojection that also requires depth information [209]. Having an image and depth map, the 3D geometry can be reconstructed, but information from different viewing angles might be missing [29]. Therefore, the viewing cone would be limited. The scene can be represented as a set of textured planes [280] or other simple objects such as spheres [42]. The depth associated with the pixels can be quantized, segmented, and processed to produce a very simple geometry. The set of billboards would result in a fast rendering; see Fig. 2.33. However, some parts of the scene might create the cardboard effect [291] due to the lack of depth-related details.

(a) Front view on the billboards looks natural.

(b) Shifted view looks natural but some 3D details are missing.

(c) Side view reveals the incomplete geometry.

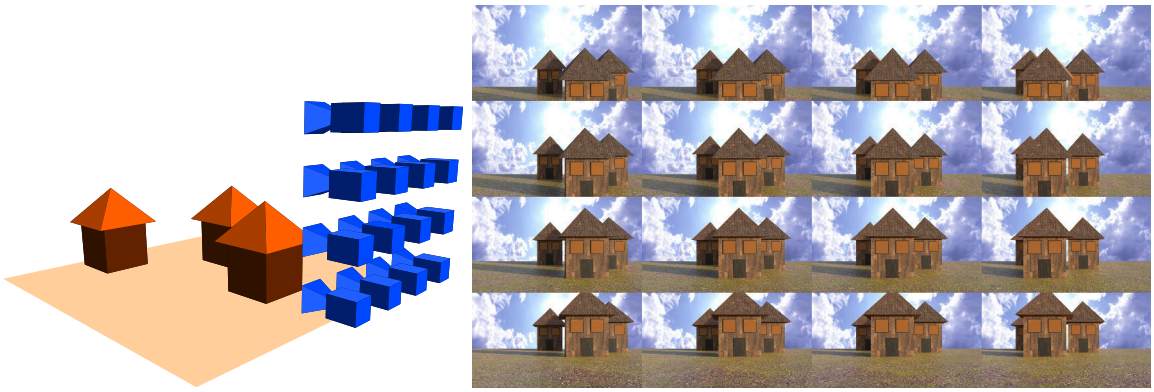
Figure 2.33: The figure shows how billboards can replace a 3D scene. They are constructed according to one viewing direction. They can produce acceptable results when viewed from this direction and slightly shifted camera positions. Similarly to light fields, they cannot be used for views from completely different camera positions.

Real-time global illumination methods often use probes as an acceleration of rendering instead of raw ray tracing of the scene [97]. The probes can store the 3D incoming light information in the form of a light field [180]. This approach can be further extended to handle dynamic lighting [53]. The geometric and radiance information can be separated and the radiance reconstructed using the principles of deferred shading.

Real-time image-based light field rendering methods are potentially faster and less memory demanding than scene reconstruction and rendering of the result because of a reduced number of necessary operations. 3D reconstructions might also fail completely when the necessary correspondences between the views are not found or are not distributed properly in the 3D space. Light field rendering always produces at least the best-effort result, where the scene is reconstructed with visible artifacts in the worst case.

2.5 Capturing of Discrete Light Field

The discrete approximation of light field, in the form of multiple images, can be captured by a grid of cameras as shown in Fig. 2.34. The grid can be replaced with one moving camera in the case of static scenes. The shape of the grid depends on the rendering method used. The grid is generally rectangular and planar with constant spaces between the cameras [113, 237, 238, 267].



(a) Camera array captures a scene. (b) Resulting views are stored as images.

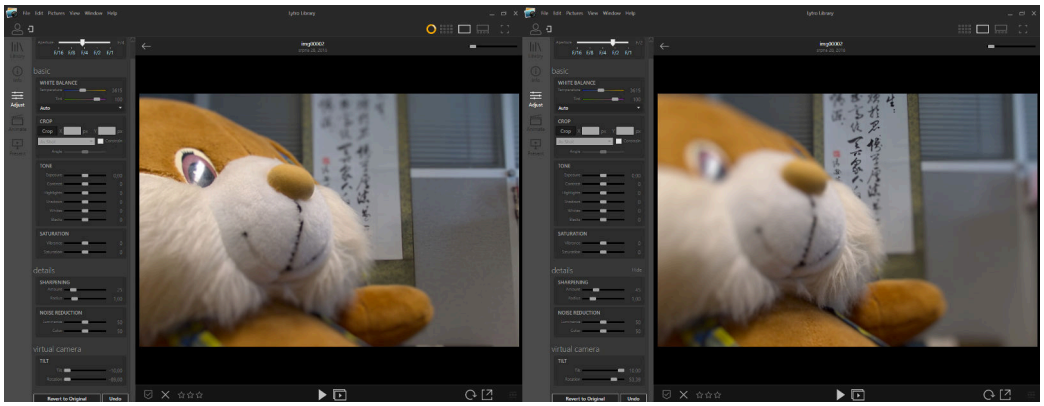
Figure 2.34: The figure shows a 4×4 array of cameras in a synthetic scene and the corresponding grid of captured views.

Another way to capture light field data is by using a plenoptic camera [220]. The set of images is acquired in the same way as with a camera grid except that only one sensor is used and the incoming rays pass through a microlens (lenslet) array that is between the main lens and the sensor [195]. Lytro company [87] had released three models of plenoptic cameras, where Lytro Illum became the most popular and capable of acquiring 4 megapixel views. However, the extracted views from such cameras are very close to each other, so the main use-case for such photos is only the refocusing and related effects. Fig. 2.35 contains three handheld devices that can be used to capture light field views. The resulting photos can be viewed in Fig. 2.36.

Figure 2.35: Three devices able to capture light field data are displayed in the photo. It is a standard Samsung Galaxy NX camera with 3D stereo lens, Lytro plenoptic camera, and 3D stereo camera on Leia LumePad 3D tablet.

(a) 3D lens provides a small displacement.

(b) 3D tablet also provides close stereo images.



(c) Plenoptic camera obtains lots of close views used for refocusing.

Figure 2.36: The results of the 3D lens and the 3D tablet are simple stereo images displayed in the figure as side-by-side views. The plenoptic camera stores a plenoptic image that can be refocused or tilted, as shown in two different views in the official Lytro Desktop software.

Camera grids have the advantage of being scalable, having a larger distance between the cameras, and potentially having higher resolution because each view is captured by the whole sensor. A similar solution targeting industry and science is Raytrix 3D Camera [211]. Plenoptic acquisition principles are also used in Fourier integral microscopes [235]. Multiple sensor cameras can be used to increase resolution and reduce noise in standard photography [224]. This approach is often used in modern smartphones to widen the effective image sensor area for better zooming and telephoto options [34]. A two-sensor camera is used in the LumePad 3D tablet to acquire a stereophoto that can be viewed directly on the device [78]. 3D information can also be captured with the standalone depth sensor. An example of this approach is Microsoft Azure Kinect [298].

The distance between views is an important parameter that affects the quality of the resulting image. The closer the images are to each other, the fewer artifacts occur in the final render because the differences between images are very small. The 3D effect is less visible if the angular resolution remains the same while the distance between the cameras is decreased. The sampling rate of light fields during acquisition depends on the content of the scene and the final use case of the rendering. Optimal sampling can be estimated by spectral or optical analysis of the scene [49].

A study, conducted by the author of this thesis, revealed that input light field images captured in logarithmic color profiles can improve the visual quality of the rendering methods [7]. Cameras offer various capturing color profiles that map the input light intensity into digital color values by different functions. Synthetic scenes can also be rendered in such profiles. The logarithmic profile CanonLog3 is the most efficient. Logarithmic profiles reveal details in areas that are too dark and too bright [98]. They are often used for artistic purposes as they enhance the post-processing options.

2.6 Compression of Discrete Light Field

One of the main drawbacks of light field rendering are the high memory space requirements. Light field data are often larger than standard 3D geometrical representation. The capturing cameras usually record the same geometry with a lot of redundancy between the views [49]. The memory requirements on the GPU can be reduced by efficient data reuse approaches and progressive rendering [154]. Efficient compression methods are still necessary and can be used even in combination with smart streaming of the necessary parts of the light field grid. Compression methods can also improve the visual quality of the novel view rendering. If a highly efficient compression is used, the light field might be captured extremely densely without large spaces between the views. Closer views are easier to interpolate If the density of the views is larger than the necessary sampling density, the novel view rendering might not even be necessary.

2.6.1 Standard Methods for Light Field Compression

The performance of the main image coding standards, JPEG, JPEG 2000, H.264/AVC intra profile, and H.265 intra profile was evaluated [17]. The *intra* suffix refers to the fact that the individual views were compressed independently (intra profile). The H.265 intra profile proved to be the most efficient compression method.

Three strategies using the H.265 were also compared [268]. The first strategy performs compression directly on the lenslet image. Another strategy arranges 4D light field views in spiral order and subsequently compresses them. The last strategy compresses a subset of lenslet images through the transformation to 4D light field. The results show that coding 4D light field leads to better performance compared to coding the lenslet images directly.

The performance of JPEG, JPEG 2000, and SPIHT (Set Partitioning in Hierarchical Trees) directly on lenslet images was also compared [110]. The JPEG 2000 exhibits the best compression performance.

4D light field views can be rearranged as tiles of a large rectangular image [208]. This image is then compressed using the JPEG 2000 coder. The proposed scheme was compared against standard image coding algorithms, namely JPEG 2000 and JPEG XR. However, it is unclear how these standard coding algorithms were exactly applied to the 4D light field data.

The similarity between light field images is high. Fig. 2.37 shows a Farneback dense optical flow calculated between two views from the light field grid. The motion vectors correspond to the structure of the scene and identify a lot of redundant data.

(a) First image is taken. (b) Second image is taken. (c) Optical flow shows a motion.

Figure 2.37: The dense flow shows that similar pixel blocks are correctly detected between the neighboring images in light field grid. That suggests that compression methods exploiting this correspondence might be efficient.

4D light field can also be rearranged into a three-dimensional body [11]. The three-dimensional volume is then encoded using the 3D DCT scheme on 8×8×8 blocks, similarly as in the JPEG coding system. Fig. 2.38 shows how light field views can be rearranged for different compression methods.

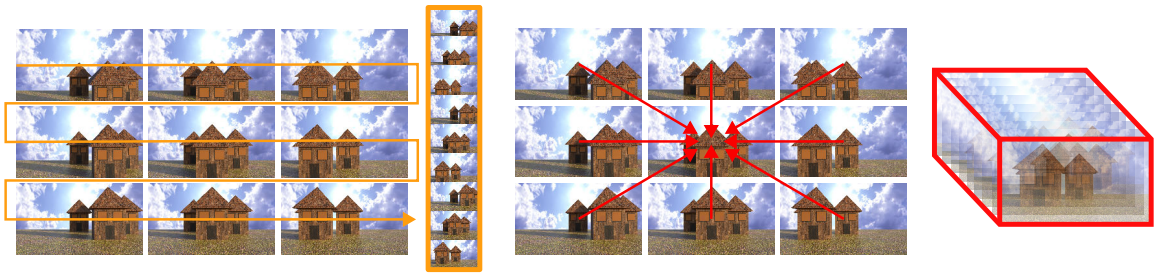


Figure 2.38: Two main approaches are used in light field compression. The views are rearranged from the grid to a pseudo-sequence which is encoded, for example, using video encoders, as depicted on the left image. The views can be also aligned to form a volume which can be compressed by exploiting the redundant parts which overlay each other, as depicted on the right image.

H.264 standard describes the Multiview Video Coding (MVC) extension, which is also described in the H.265 (HEVC) standard (MV-HEVC) for optimal coding of animated multiview sequences [72]. This extension is capable of encoding multiple views and exploiting their redundancy. It was designed for stereoscopic videos. The main downside of these extensions is the lack of implementations, especially in open-source software and libraries. Hardware acceleration is almost non-existent. It was used as part of the light field compression scheme with hierarchical organization of the views [12]. An efficient 3D prediction structure for MV-HEVC was proposed [141] and proved to be better than similar previous approaches [271]. MV-HEVC can be easily adjusted to support the light field grid and several prediction schemes were compared [21]. The results show that MV-HEVC can be used for a random-access encoding which is suitable for light field data.

2.6.2 Video Methods for Light Field Compression

Light field compression with video codecs outperforms other 2D or 3D extended image compression methods according to previous research [12, 1, 2, 228]. Another reason for the promotion of video codecs is their widespread usage, the variety of available implementations, and the increasing support on modern GPUs [202] with the new AV1 support on the NVIDIA® RTX™ 3xx series and possible future support of current state-of-the-art XVC [226] and VVC [39] (Versatile Video Coding) formats. H.265 standard [251], similarly to alternative formats, offers intra-frame coding, which compresses the frame independently. This approach uses techniques similar to other image compression algorithms such as JPEG [270], for example, block-based processing that exploits spatial redundancy in the image, quantization of data in the frequency domain or chroma subsampling.

Inter-frame coding, available in all standard video compression methods, exploits temporal redundancy. Similar blocks in multiple frames can be identified using block-matching algorithms [245] and encoded as blocks with motion vectors instead of multiple blocks. The input video frames, shown in Fig. 2.39, are compressed in three basic ways [33]:

- **I-frame** is a keyframe that is encoded only by intra-frame methods. It can be decoded without information from any other frames.
- **P-frame** is a difference frame which encodes only changes (new parts of the image and motion vectors) from the previous frames. It is necessary to have the relevant reference frames already decoded.
- **B-frame** is similar to the P frame but can also use later frames as reference for decoding.

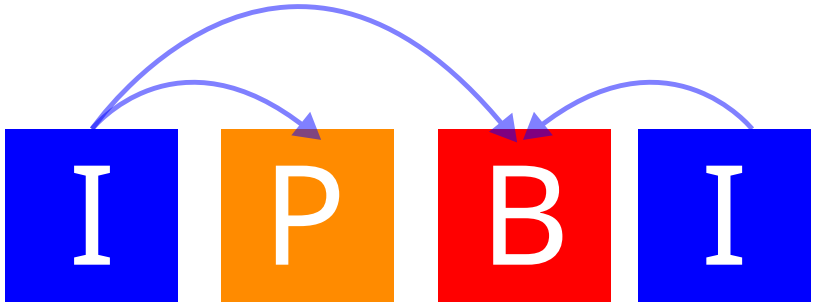


Figure 2.39: Three standard types of frames used in video compression are depicted in the figure. I-frame can be encoded separately, P-frame depends on the last I-frame, and B-frame depends on I-frames on both sides.

The encoded video stream is a numbered sequence of packets, where usually one packet corresponds to one frame. Modern encoding standards can also split frames into macroblocks or slices, where each slice can be independently encoded as one of the three mentioned types [234]. Group of pictures (GOP) is a sequence of encoded frames beginning with an I-frame. The rest of the frames are P or B-frames that use one or more frames from the same GOP as references. When random access in a video stream is needed, the closest GOP beginning is found. The subsequent frames are decoded until the desired one is reached. Light field data can be encoded as a pseudo-sequence that is designed for optimal compression of the whole grid [181]. Another pseudo-sequence approach exploits the redundancies in sub-aperture image intra spatial, sub-aperture image inter-view, intra-micro-image, and inter-micro-image space [187]. Neural video compression methods can also be used instead [161, 293].

2.6.3 Specialized Methods for Light Field Compression

One of the first proposals [158] was to use vector quantization to reduce redundancy between views. 2D or 4D tiles of the light field are encoded as vectors and saved as references to a codebook constructed during the training phase. This compression is lossy because the closest value in the codebook is assigned to the input vectors. A training-based dictionary technique [184] provides a way to compress light field images and videos. This approach uses a multidimensional dictionary ensemble instead of single 1D dictionaries used in different proposals. Simple approaches such as selective subsampling of the views in the grid can improve the compression ratio for many light field encoders. This approach is mainly efficient with dense light field grids [55].

Another approach divides the 4D light field into blocks and then converts to the YUV color space with downsampled chrominance [179]. Evenly distributed I-frames are chosen

from the input images and compressed using block-based discrete cosine transform and coefficient quantization. P-frames that are used as difference frames between the closest I-frames are then calculated using the disparity between the compressed image and the adjacent I-frame. Because the shift between images is known, the disparity direction can be exactly defined.

Hierarchical motion-compensated compression [212] was proposed as an efficient solution for interactive rendering. This method analyzes the relative motion between input images and applies image transformations to minimize redundancy in a hierarchical manner. Both motion compensation and hierarchical approaches are combined into one hybrid method. The main advantage of this approach is a fast GPU-friendly decoding. Similarly to other methods, this method loses efficiency as the distance between the input images increases. Fig. 2.40 shows an example of the identification of common elements in light field.

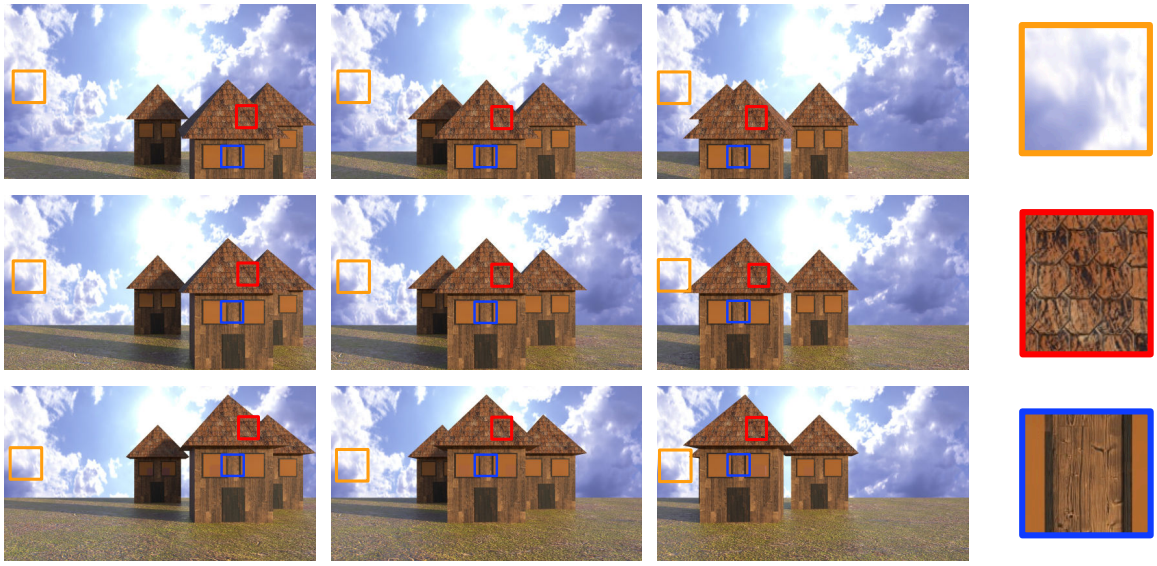


Figure 2.40: Blocks which contain the same visual information in all light field views can be detected. These blocks can be used in dictionary-based compression or for inter-view prediction. Note that simply identifying blocks of pixels in multiple views that come from the same spot in the scene is not enough. Reflections and other lighting effects produce view-dependent changes in some materials.

Other standard image or video compression methods are used directly or in hybrid approaches, such as displacement intra prediction [165]. The image displacement in the 4D light field structure is estimated using the intra prediction of the H.265 standard in both the time and spatial domains.

Sparse coding scheme for the entire 4D LF based on several optimized key views can be used for an efficient compression [52]. Another method decomposes the 4D light field into homography parameters and residual matrix [130]. The matrix is then factored as the product of a matrix containing k basis vectors and a smaller matrix of coefficients. The basis vectors are then encoded using the H.265 intra profile. Similarly, H.265 can be used to remove redundancies in dynamic mode decomposition compression [215]. A hierarchical coding structure for 4D light fields can also be used [163]. The 4D LF is decomposed into multiple views and then organized into a coding structure according to the spatial coordinates. The scheme is implemented in the reference H.265 software. Another approach

is a coding scheme that splits the 4D light field into several central views and remaining adjacent views [103]. The adjacent views are subtracted from the central views and then both groups are encoded using the H.265 coder. The 4D light field can be fed into the H.265 exploiting the inter-prediction mode for individual LF views [166]. Disparity-based sparse prediction [20] warps the reference views to the compressed ones with normalized disparity maps. Then it reconstructs the view with inpainting and filtering based on the result. JPEG2000 is used to encode the disparity views and H.265 to encode the residual and color views. This method can be used for both dense and sparse datasets and outperforms alternative plenoptic and video encoders.

A lot of research focuses on compression based on convolutional neural networks [116, 125, 299]. The learning process might often be time-consuming. Deep image prior [157], with random neural network initialization, can be used to compress light field data without the training process [129]. It is compact and lightweight and can reach better compression ratios than alternative networks. All training-based methods depend on the quality of the training phase and are usually not suitable for memory-efficient and fast GPU decoding.

Various compression schemes have previously been proposed for different light field formats or live-streaming scenarios [146]. Data streaming can be optimized by analyzing the current light field coverage of the scene so that only the data necessary to improve the result are uploaded to the memory [74].

Light field rendering and compression are not just connected as different steps of the light field workflow. Novel view rendering and synthesis can also be used as a compression method. The desired views can be synthesized from other existing views similarly to reference and prediction frames in video compression methods but with fewer information about the novel view. Deep learning can be used to estimate the 2D view from the sparse sets of 4D views [24]. A sparse set of images is encoded using a hybrid video encoder, and the missing views are approximated from the decoded ones. Similarly, multi-disparity geometry structure estimated from a sparse set of images can be used in a multi-stream dense view reconstruction network to synthesize a novel image for decompression purposes [171]. This method handles well the occluded geometry and reveals texture details in high quality. Residual CNN-Assisted JPEG compression method [106] outperforms the GPU-accelerated H.265 decoding speed 10× for light field compression. The method uses depth-based warping of the central image from Lytro camera scenes. The central image is encoded in JPEG format. A combination of novel video compression format VVC and deep learning for missing view synthesis uses advantages of both approaches [23].

Disparity-assisted models can be more useful in light field compression due to the spatial distribution of information across the grid [241]. However, novel standard image/video compression methods can still reach results close to the schemes tailored for light field. Especially when the views are far from each other and not captured by a plenoptic lenslet camera, the disparity-based warping can introduce a significant amount of artifacts.

2.7 GPU-Accelerated Decompression of Discrete Light Field

The following sections contain a deeper description of a HW accelerated decoding method [5] suitable for the main proposals of this thesis. These methods are the author's original extension of the state-of-the-art field. Existing compression standards were investigated, and, based on the results, a novel compression scheme was proposed. These experiments and methods are not claimed to be the main scientific contribution of this thesis but are necessary to support the main proposal. Acceleration of decompression is important in light field rendering because the decompression process might be utilized several times during each rendering time frame.

2.7.1 Evaluation of Light Field Compression Methods

This section is based on two published papers [1, 2]. The impact of state-of-the-art image and video compression methods on the quality of images rendered from light field data was evaluated. The results presented here serve as motivation for the usage of video encoding in the following light field compression scheme. The main question was: *Which of the widely used state-of-the-art image/video compression methods is the optimal choice for light field data?* Four-dimensional light field data can be compressed much more than independent still images while maintaining the same visual quality. The video compression methods seem to be an efficient solution for light field compression due to the best compression ratio and wide support in the existing software and hardware.

Both main light field formats are included in the measurements performed. Light fields acquired by a single compact sensor have limited support for the viewing angle. Light fields based on the array of cameras offer larger viewing angles at the cost of missing information in between the cameras. Considering increasing resolution sensors, it is no surprise that the light field data reach huge sizes. For example, a scene from the standard dataset [267] is captured using a 17×17 grid of cameras with image resolution 1536×1152 (rectified and cropped). The uncompressed size easily exceeds one gigabyte.

Dataset

The first group of measured methods covers the image coding methods JPEG and JPEG 2000. They are sometimes referred to as self-similarity-based methods [162]. The second group comprises the video coding methods H.265, AV1, VP9, XVC, and VVC. These methods are referred to as pseudo-sequence-based methods. The third group consists of JPEG 3D custom extension and JPEG 2000 3D (Part 10, JP3D). The fourth group extends the image-coding methods into four dimensions with a custom JPEG 4D extension. To evaluate the above methods, the following list of encoders was used: OpenJPEG[2], x265[3], libaom (AV1 Codec Library)[4], libvpx (VP8/VP9 Codec SDK)[5], Divideon xvc codec[6], Fraunhofer vvc codec[7] and custom implementation of the JPEG method. Tab. 2.2 describes the dataset used in the measurements.

[2] openjpeg.org
[3] github.com/videolan/x265
[4] aomedia.googlesource.com/aom
[5] github.com/webmproject/libvpx
[6] github.com/divideon/xvc
[7] github.com/fraunhoferhhi/vvenc

Scene	Source	Resolution	Disparity
Black Fence	EPFL Light-field [220]	$15 \times 15 \times 625 \times 434$	-1 to 1
Chessboard	Saarland University [237]	$8 \times 8 \times 1920 \times 1080$	40 to 90
Lego Bulldozer	Stanford Laboratory [267]	$17 \times 17 \times 1536 \times 1152$	-1 to 7
Palais du Luxembourg	EPFL Light-field [220]	$15 \times 15 \times 625 \times 434$	-1 to 1

Table 2.2: Dataset used in this study. The first and last light field are taken using a plenoptic camera; the Chessboard is captured using a camera array; the Lego Bulldozer is captured using a motorized gantry holding a camera. The adjacent image disparity range (last column) is given in pixels.

2.7.2 Evaluation

A simple comparison of compressed and uncompressed raw data is not enough to evaluate the encoding of light fields. The uncompressed data were used in a light field refocusing rendering, and the data before and after compression were compared. Fig. 2.41 describes the evaluation scheme used. A quick experiment shows a significant disparity in the methods used; see Fig. 2.42. This difference is about 10 decibels in the PSNR, depending on the bitrate and compression method. This can be explained by the fact that any pixel in the rendered view is a sum of pixels from the 4D light field. This sum suppresses compression artifacts. The 4D light fields can be compressed much more than independent images, while maintaining the same visual quality.

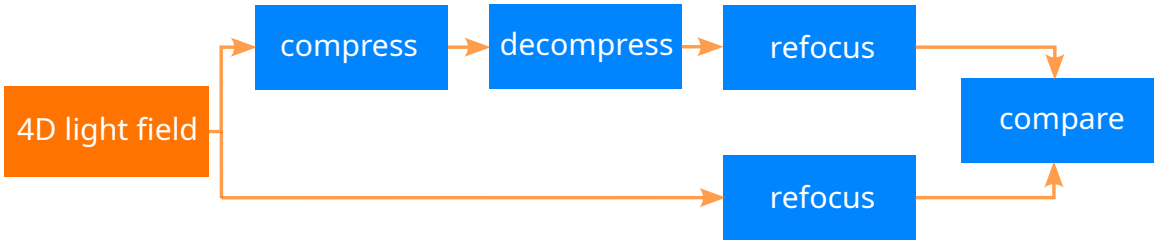


Figure 2.41: The data flow diagram describes the compression performance assessment methodology used in this study.

2D, 3D and 4D Compression

To fairly assess the difference between multidimensional compressions, an identical method must be used. A custom implementation of the JPEG compression method was created with the ability to process 2D, 3D, or 4D data. Although JPEG 2000 supports 2D and 3D data compression, it cannot process 4D images. Since the similarity of adjacent pixels in the third and four dimensions strongly depends on the camera baseline, different results can be expected depending on the baseline distance. The result of this experiment is shown in Fig. 2.43.

3D compression methods clearly outperform their 2D counterparts on light fields with a small baseline (*Black Fence* and *Palais du Luxembourg*). 4D JPEG also outperforms its 3D counterpart. Pixels at the same spatial position in adjacent views are strongly correlated. However, the situation changes with increasing baseline (*Lego Bulldozer* and *Chessboard*). Adjacent views are less and less similar, resulting in higher amplitudes of the underlying

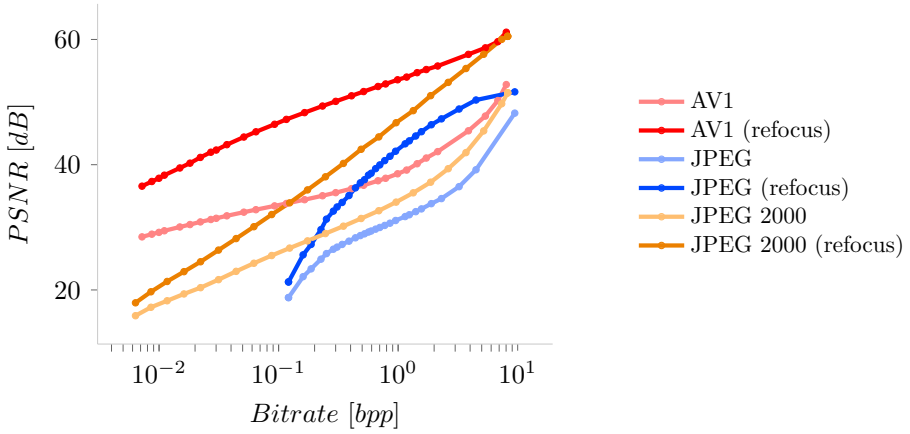
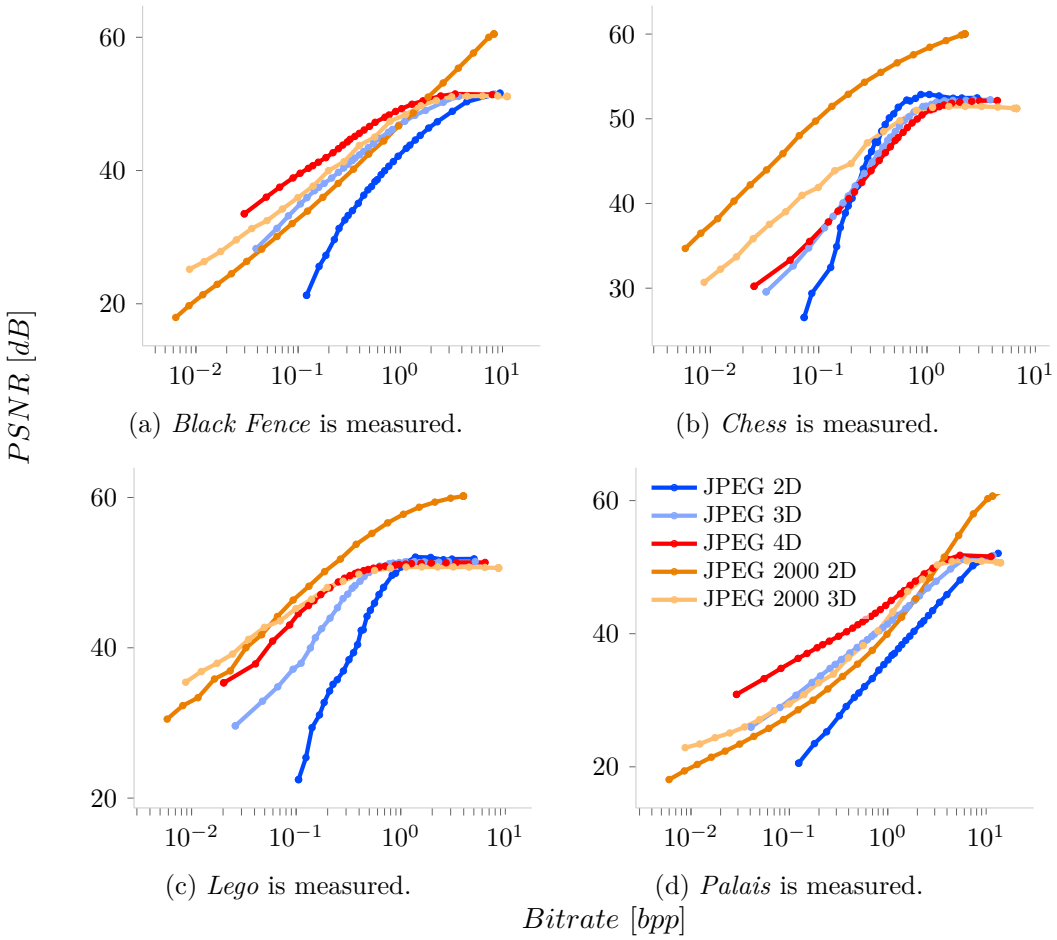


Figure 2.42: The chart shows the difference in the quality assessment using the 4D light field directly compared to using images rendered at virtual focal planes. The results are measured with the *Black Fence* scene.



(a) *Black Fence* is measured.

(b) *Chess* is measured.

(c) *Lego* is measured.

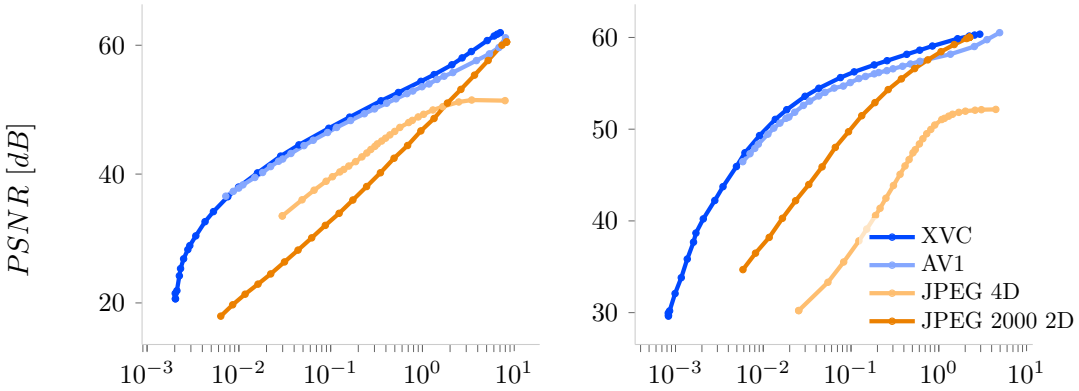
(d) *Palais* is measured.

Figure 2.43: Image compression methods are compared against their extensions into three and four dimensions.

transform coefficients. Consequently, the tide is turning in favor of the less-dimensional compression methods. Taking into account the JPEG method, the *Lego Bulldozer* is a special case because it contains large areas of black pixels. It turns out that it is more efficient to compress these solid areas at once using a single 4D block than using multiple 3D blocks. Similarly, it is more efficient to use a single 3D block than multiple 2D blocks.

Video Compression

4D light fields can be compressed as a sequence of 2D frames, as multi-dimensional volume, or as a video. The overall comparison is shown in Fig. 2.44. Video compression methods perform better than all image compression methods, even better than their 3D and 4D extensions.



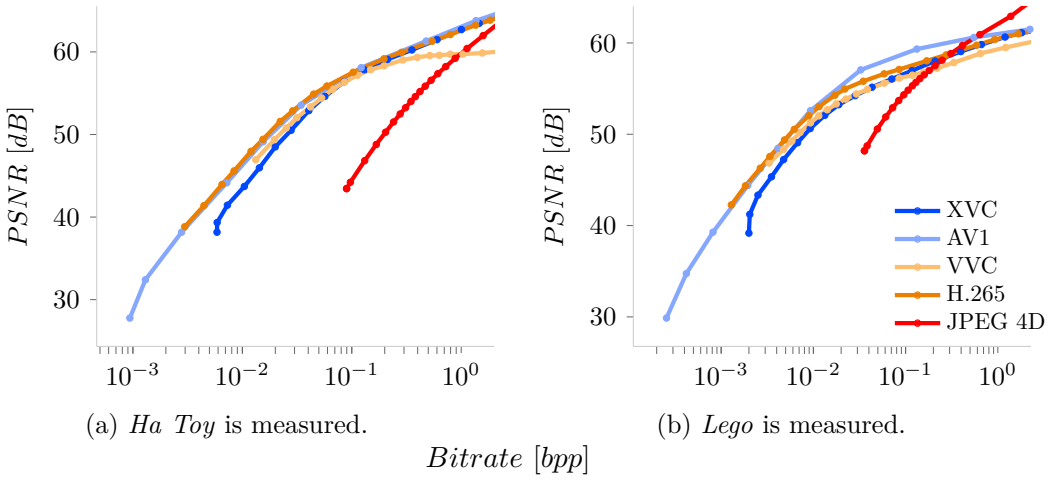
(a) *Black Fence* is measured. (b) *Chess* is measured.

Figure 2.44: Video compression methods perform better than all image compression methods according to the conducted experiment.

The video compression formats were compared in the refocusing scenario; see Fig. 2.45. Another measurement was performed to additionally evaluate the effect of compression on 3D point cloud reconstruction from light field; see Fig. 2.46. A distortion is apparent at lower bitrate, which is caused by noise and a reduced number of vertices present in the point clouds. Video compression methods show similar results with the newest standards slightly outperforming the old ones. These methods might be an optimal way to compress light field data with a wide support in current software and hardware.

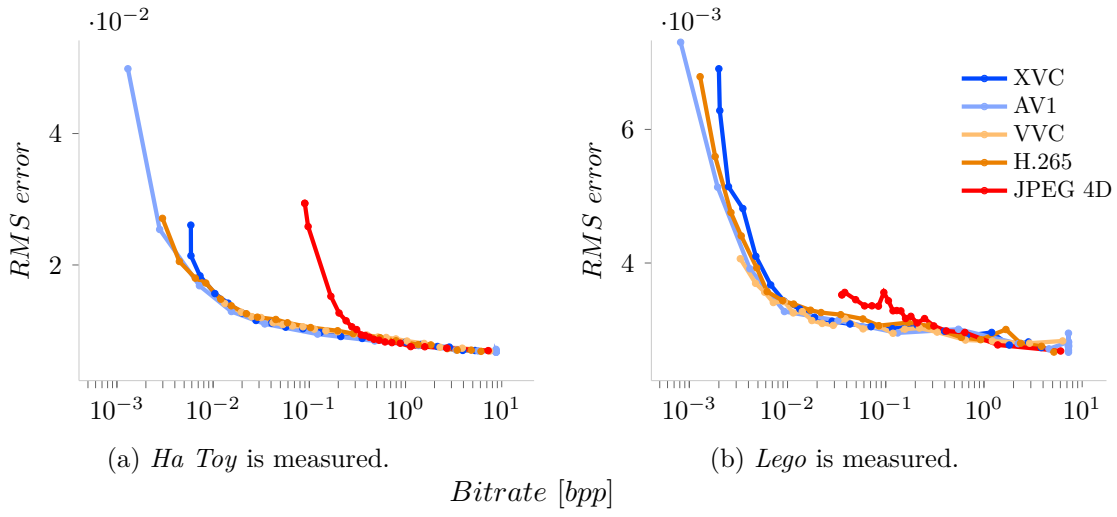
2.7.3 Efficient Video Decoding Method

The proposed method belongs to pseudo-sequence-based video compression where light field views are used as video frames; see Fig. 2.47. One selected frame serves as a fully encoded standalone reference, and other frames are encoded as motion-compensated differences from this reference. The proposed compression is described with a 1D light field grid, where the views are captured along a horizontal line. This method can be easily used for other shapes of the light field grids as well.



(a) *Ha Toy* is measured. (b) *Lego* is measured.

Figure 2.45: Different compression formats are compared on views rendered for multiple focal planes.



(a) *Ha Toy* is measured. (b) *Lego* is measured.

Figure 2.46: Different compression formats are compared on 3D point clouds reconstructed from 4D light fields.

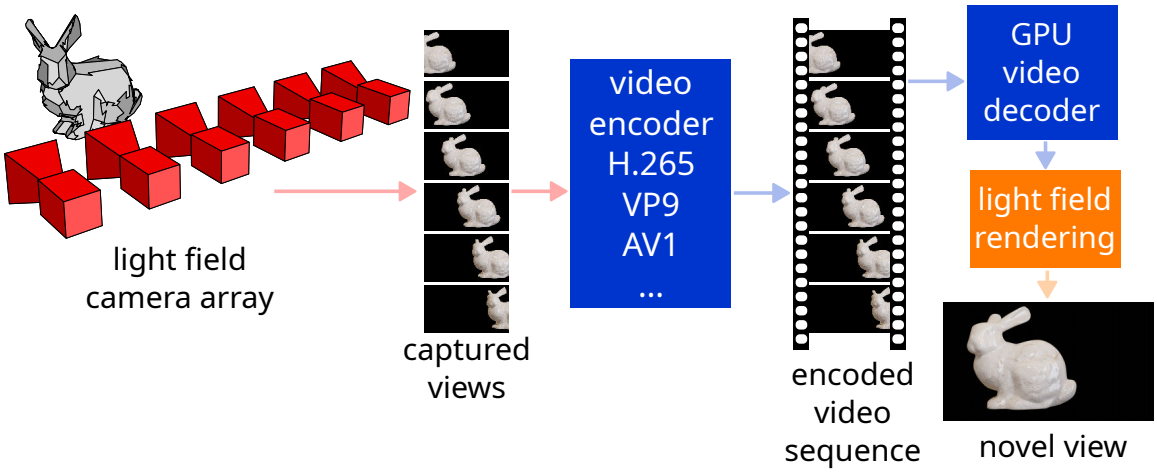


Figure 2.47: The scheme describes the video encoding principle used for efficient data streaming to GPU in light field rendering.

Frame Type Sequences

Ideally, only the necessary compressed video packets are uploaded to the GPU memory and decoded there; see Fig. 2.48. The host and GPU memory data transfer is the bottleneck of light field streaming. Using the classic encoding scheme, all previous packets from the GOP need to be decoded before the desired frame is acquired. If the light field views are encoded row-by-row, then multiple GOP jumps would be necessary to decode four nearest views to the novel one. The interval between I-frames (size of GOP) can be shortened. The data size would be larger because of more encoded space-demanding I-frames, but the desired frame would potentially need fewer previous frames decoded. The shortening of GOP can reach the limit, so that all frames are encoded as I-frames and can be decoded without other frames.

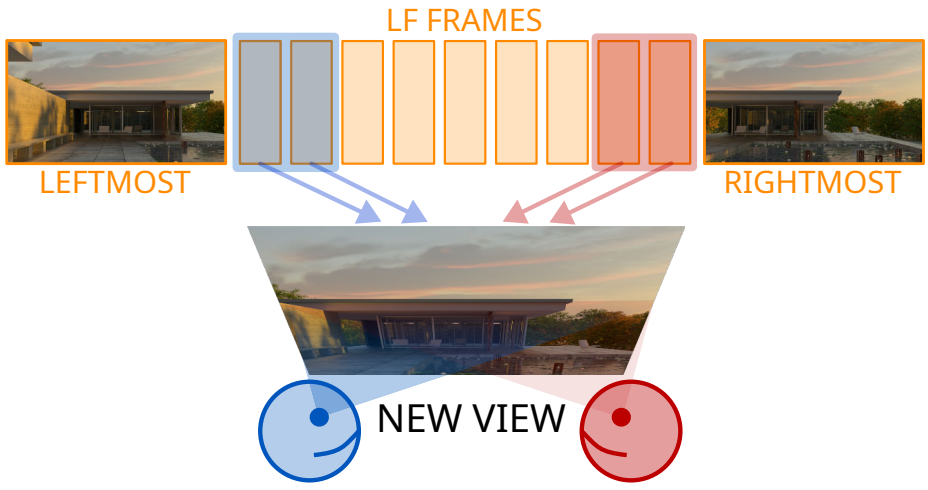


Figure 2.48: Only a subset of the light field views is necessary for interpolation of a novel view. The needed frames are determined by the viewing position of the user.

Compression Scheme

The proposed optimal scheme for light field data encodes an I-frame in the middle of the sequence. All other P-frames are encoded using only this initial I-frame as a reference. This approach would not be efficient for arbitrary video, because of huge differences between its different parts, especially when the scene changes. However, light field views are very similar to each other because they capture the same static scene, and the distance between the views is usually not significant enough to create large differences between the images. Light field rendering methods often exploit this similarity. After decoding the initial I-frame, all other frames can be decoded simply by uploading their packets to the decoder. The minimum number of packets to decode a frame is one I-frame packet and one P-frame packet, but only one P-frame packet is needed to decode further frames. The decoding speed might also be faster because the decoder only needs to read the data from one packet and also internally from one I-frame to decode the new frame. The encoding schemes that are compared are:

- **Proposed** with one I-frame in the middle of the sequence and the rest P-frames dependent on the one reference.
- **One-GOP** and **GOP30** are classic GOP schemes, one I-frame at the beginning, and the rest are P-frames sequentially encoded in a classic way with possible multiple dependencies and a version with I-frames enclosing every 30 frames.
- **All-I** where all frames are individually encoded as I-frames without any dependencies.

See Fig. 2.49 that compares the described schemes.

(a) All preceding frames need to be decoded sequentially in One-GOP or GOP30.

(b) The desired frame can be decoded directly in all-I.

(c) One reference frame needs to be decoded before the desired frame in the proposal.

Figure 2.49: P-frames can be decoded only after all their reference frames. I-frames can be decoded without any other frames. **Fourth** frame is the desired one in this example.

The main reason why light field assets are not commonly used instead of 3D mesh geometry is their space requirements. One option is to keep the entire asset in the GPU memory. The decoding and rendering would be relatively fast, but usually 3D simulations or games require a lot of GPU memory. Storing a large number of assets like this might be inconvenient. The second option is to stream the needed views of the light field asset in the GPU memory on demand. The views need to be decompressed on the CPU and transferred via PCI-Express in GPU memory. The importance of compressed data transfer and GPU decoding has been increasing recently, as reflected in relevant proposed solutions, such as DirectStorage[8] by Microsoft, which allows streaming of compressed multimedia files while decoding them with GPU shaders. The workflow for the proposal is illustrated in Fig. 2.50.

[8] devblogs.microsoft.com/directx/directstorage-is-coming-to-pc

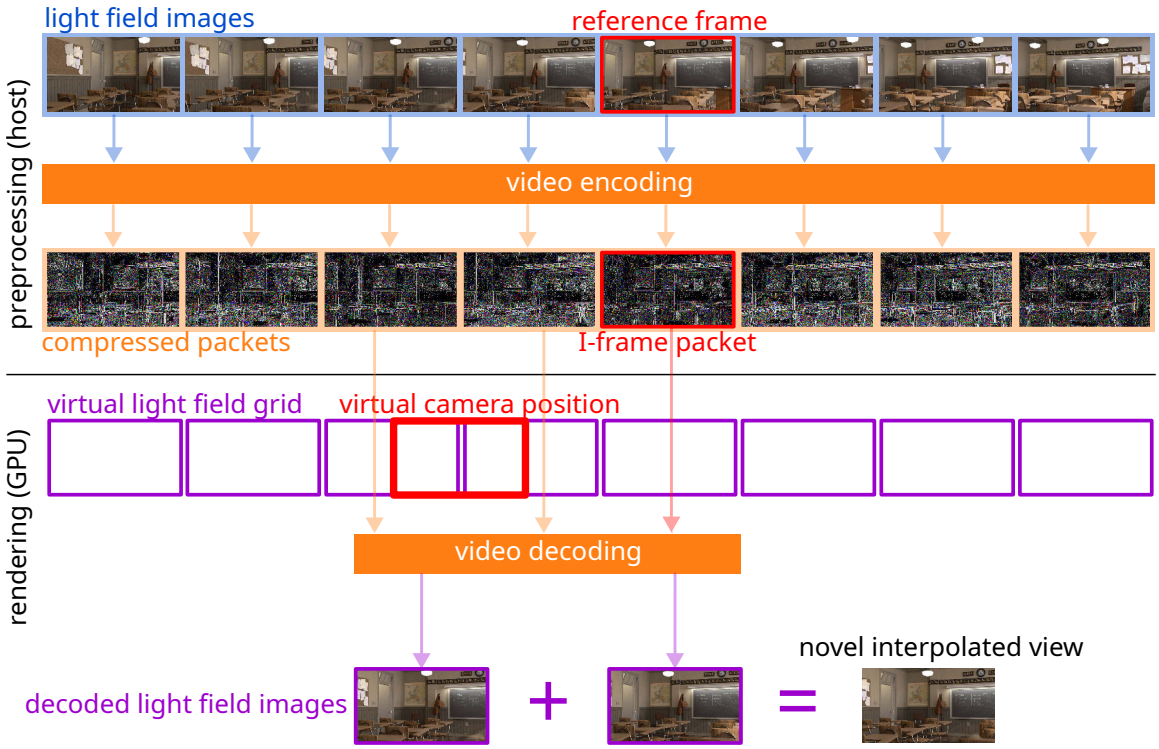


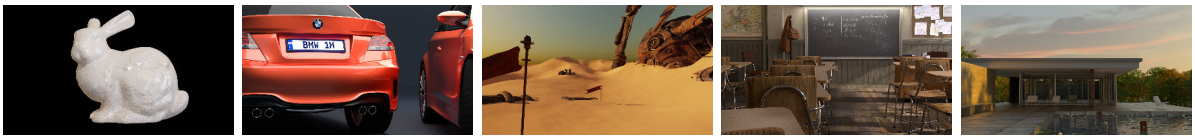
Figure 2.50: The input light field images are encoded with the proposed compression scheme with one I-frame. The rendering method identifies the necessary views according to the virtual camera position. Usually, only the closest views are necessary. These views are decoded and only their packets with the I-frame are transferred to GPU. The packets are decoded on GPU and used to interpolate a novel view.

2.8 Decompression Evaluation

Measurements were performed on a machine equipped with GeForce RTX™ 2070 GPU and Intel® Core™ i5-8500 CPU 3.00 GHz CPU, running Arch Linux. The relative comparison should be the same on other hardware (HW) and should not depend on the used configuration.

2.8.1 Dataset

A custom dense (500 4K views) 1D light field dataset was rendered and used in the experiments; see Fig. 2.51. The new dataset was created to reflect the current 4K video standard and also to cover extremely dense data. Previously used high-resolution light field datasets are usually sparse [267]. The 1D dataset simulates a horizontal 3D motion. It can be suitable for views through virtual windows, expecting the camera to move at almost constant height from the floor. Another use case is on holographic 3D displays, such as by Looking Glass Factory, that support only horizontal view change, expecting the user to be in a sitting position.



(a) *Bunny* contains one simple object with static background.

(b) *Cars* contain reflective objects with simple background.

(c) *Sand* contains a landscape with small number of features.

(d) *Class* is a complex indoors scene.

(e) *Pavilion* is a complex outdoors scene.

Figure 2.51: Center views of the five scenes in the used dataset are shown in this figure. The scenes cover the usual settings of light field data.

The dataset is used in three versions: full, every fifth, and every tenth view. The greater distance between the views and the smaller view count might affect the compression efficiency and also the light field rendering methods and their sensitivity to compression artifacts. The scenes were taken from Blender demo projects[9] where the *Pavilion* scene is modelled by an external artist[10] and the *Bunny* 3D model is a classic Stanford Bunny[11].

2.8.2 Streaming Capabilities

A simple experiment was conducted to compare the image decoding capabilities of modern GPUs. Fig. 2.52 shows how H.265 (implemented using NVIDIA NVDEC via FFmpeg[12]) outperforms JPEG2000 (implemented using nvJPEG2000 library[13]) when decoding a sequence of 4K light field views on GPU. H.265 has a physical HW acceleration unit on modern GPUs and also reaches a better compression ratio for light field data due to intra-frame redundancy, as shown in Fig. 2.53. GPU-accelerated texture compression formats

[9] blender.org/download/demo-files
[10] Hamza Cheggour - emirage.org
[11] graphics.stanford.edu/~mdfisher/Data/Meshes/bunny.obj
[12] trac.ffmpeg.org/wiki/HWAccelIntro
[13] developer.nvidia.com/nvjpeg

are inferior to the more advanced compression methods and are not suitable for the task. Video formats seem to be the best choice for streaming light field data on a GPU.

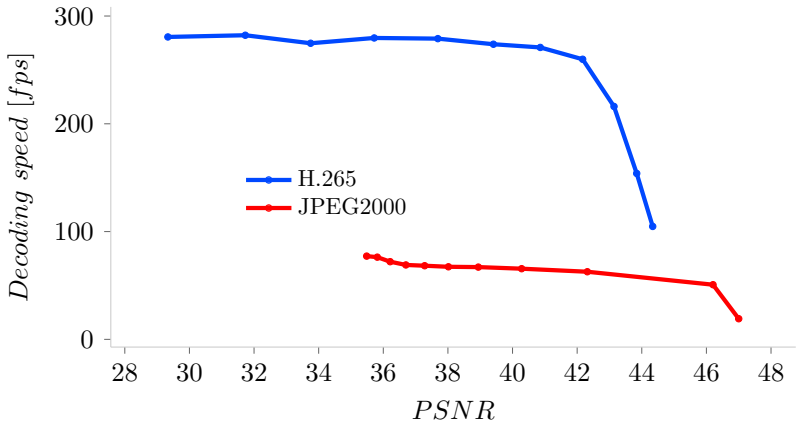


Figure 2.52: GPU-accelerated H.265 and JPEG2000 format decoding is compared in this chart. 50 frames of *Pavilion* dataset were encoded with different quality settings and decoded on GPU 50× in a loop.

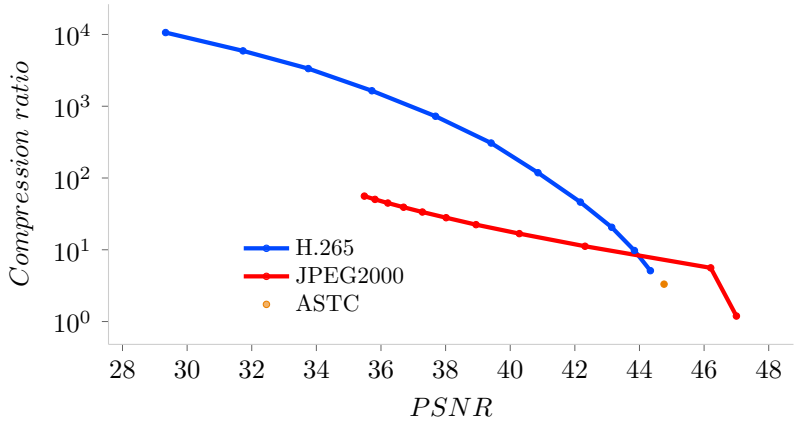


Figure 2.53: The figure shows a compression ratio comparison for H.265 and JPEG2000 formats on *Pavilion* dataset. ASTC (Adaptive scalable texture compression) GPU accelerated texture compression format is added as a reference to show, how such formats relate to the other methods.

2.8.3 I-Frame Positioning

It is expected that the user would be more interested in views near the center of the light field than at the borders. The scene is usually captured so that the area of interest is in front of the central cameras, which are often used as reference views for further synthesis [264, 267, 290]. The central view was also previously proposed in other works due to the symmetric properties of the grid [141, 181]. Artifacts in light field, for example, in a window of a virtual scene, would not be easily visible from the extreme viewing position at the edges, and even widely used image-based methods [68] do not work well in such cases. The quality of the decoded views should be the best around the middle because they are the most similar to

the reference I-frame. An experiment was conducted to compare the positioning of the I-frame in the proposed scheme. The *Bunny* dataset was encoded with the proposed scheme, which has the reference frame in the middle of the sequence and at the beginning. The result of the light field rendering quality using such decoded frames is shown in Fig. 2.54. As an alternative, saliency detection methods [83] might also be used to identify the area where the best quality is necessary.

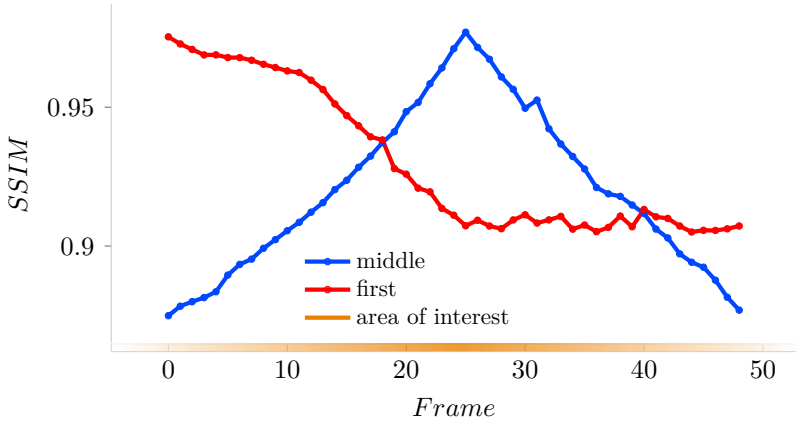


Figure 2.54: The positioning of the I-frame and its effect on the visual quality of the light field per-pixel rendering method is shown in the figure. The I-frame is placed at the beginning and in the middle of the sequence. The experiment was measured on *Bunny* dataset.

2.8.4 Decoding Time

The speedup of the proposed scheme could be even more significant when decoding four frames in the 2D light field, since only one I-frame plus four P-frames would be uploaded. The I-frame would be placed in the middle of the grid, similarly to the proposed scheme. In addition, any subsequent decodings would not require the decoding of the first I-frame, which would be kept as a reference. The proposed scheme is more efficient than encoding all I-frames in the stream. This was proved in an experiment, where the I-frame was uploaded and decoded only once; see Fig. 2.55. The decoding time t of a frame encoded at position pos encoded with constant a GOP size in the classic scheme can be roughly estimated by Eq. 2.7 assuming a simplified scenario with constant decoding time t^I of I-frames and t^P of P-frames and no B-frames. The function mod() performs the standard modulo operation.

$$t = t^I + \text{mod}(pos - 1, GOP) \cdot t^P \tag{2.7}$$

Fig. 2.56 shows the times necessary to decode one frame in the compared schemes. Note that the times for comparison of the schemes were measured including the loading time of the data from the file, uploading time of the packet to GPU and actual decoding time. Preloading in RAM would be used as an optimization in real-time applications, and the actual speed would achieve similar results as in Fig. 2.55, depending on the HW used. Having the encoded packets already in the GPU memory would also be possible due to the reduced size of the data, and the proposed scheme would still speed up the decoding.

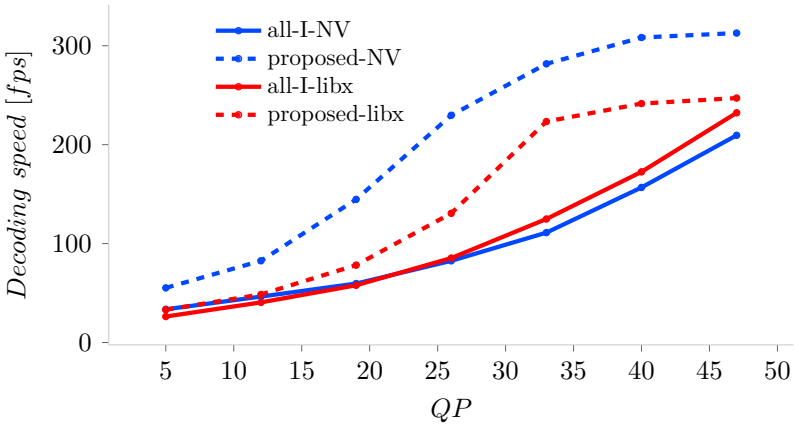


Figure 2.55: The speed of H.265 GPU-accelerated decoding was compared for the all-I-frame and proposed encoding scheme. Two H.265 implementations were used for the encoding: libx265 and hevc_nvenc. The proposed scheme outperforms the all-I in both cases. The measurement was conducted over the standardized *quantization parameter* QP.

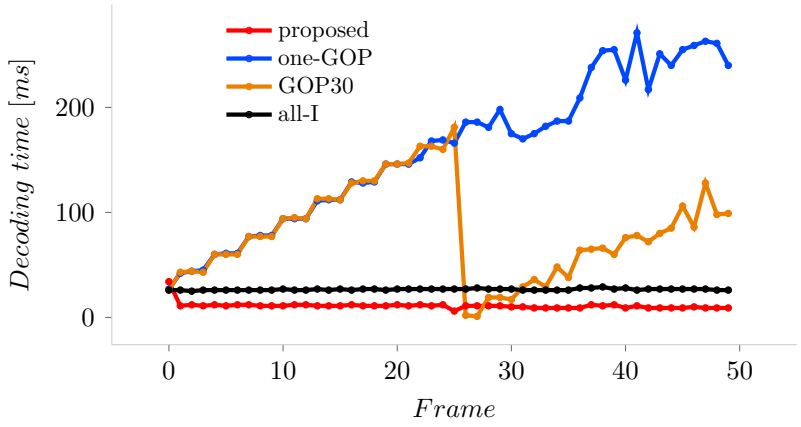


Figure 2.56: The compared schemes and decoding times for each frame were measured on *Bunny* dataset with CRF (Constant Rate Factor) set to 35. The proposed scheme has a longer decoding of the first frame due to the initial I-frame decoding. The I-frame is used directly when queried, so the decoding time in the middle of the sequence is almost zero. The jump in GOP30 marks the beginning of the new GOP.

2.8.5 Encoding Schemes Comparison

The packets are stored in one file, beginning with the I-frame. A second file contains a list of pointers in the packet file, compressed by the Deflate algorithm. The existing encoders and frameworks do not support the proposed scheme directly. The encoding was implemented in pairs, where each frame was encoded in a two-frame video with the same first I-frame and the packets were then stored separately. YouTube video platform recommends setting the GOP size of uploaded videos to half of the framerate[14]. The stream can be viewed as a 60 fps video, and the classic scheme was encoded with GOP size set to 30 frames. All videos were encoded using the libx265 encoder and decoded with GPU-accelerated NVDEC[15]. The encoder was used mainly with the default parameters. The main settings consist of: Main 4:4:4 profile, Level-5.1 (Main tier), Coding QT 64/8.

The original frames are encoded using the schemes over CRF values from 0 to 49 jumping by 7 in an H.265 stream and decoded. The decoded frames are compared with the original ones to discover how the schemes affect the compression quality. The decoded and original frames are also used to produce new synthetic views utilizing three light field rendering methods:

- **LF per-pixel** is a simplified per-pixel light field focusing method [9]. This method is the fastest, but with the lowest quality, because it is initially designed for a 2D light field.
- **LF blend** is a mix of all input images that are shifted so that a certain part of the scene is sharp and the rest is defocused as proposed in the original light field rendering method [158].
- **Deep** is a Pytorch implementation[16] of deep-learning method Super SloMo [127] for frame interpolation used in SAVFI framework [57]. This method offers the best quality but is the slowest one.

The results are compared to properly test the compression schemes in light field rendering use-case; see Fig. 2.57.

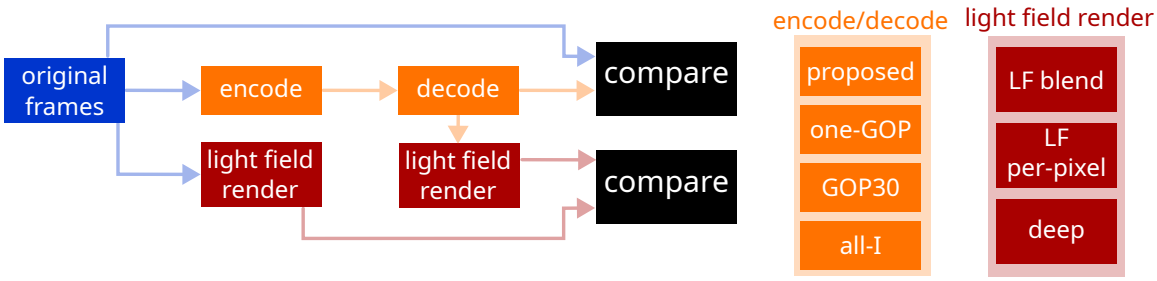


Figure 2.57: The scheme describes the evaluation. The original frames are encoded and decoded using different schemes. The decoded views are compared directly with the original frames. The decoded and original views are also used in light field rendering.

The compression ratio of the proposed scheme is expected to be worse than that of the classic one due to the way video encoders are designed. The main question is: *How much*

[14] support.google.com/youtube/answer/1722171
[15] developer.nvidia.com/video-codec-sdk
[16] github.com/avinashpaliwal/Super-SloMo

would the visual quality decrease with the proposed scheme and would the packet size still be small enough to outperform the other schemes in terms of GPU decoding speed? The objective of the measurement is to find which scheme has the best decoding time over visual quality ratio. The results presented in Fig. 2.58 show that the proposed scheme can be used in light field rendering methods to achieve real-time performance, faster and at the same quality levels as the alternate schemes. Note that the measurement performed decoding of the frames one by one over the whole dataset. When streaming continuously, the decoding times might be even shorter, similar to Fig. 2.52.

The proposed scheme is not the best choice for storing light fields on the disk because of the poorer quality/size compression ratio. However, the proposed scheme significantly outperforms other schemes in decoding speed and is the best in all measurements with the quality/speed ratio. To further investigate the results, two additional metrics were measured.

The first *error* metric in Eq. 2.8 to compare the average results takes into account errors in relation to the best values of the three measured quantities: *visual* quality, decoding *time* and the resultant stream *size*. Visual quality error is usually small. Therefore, it is amplified to be more significant. The proposed scheme has the lowest error metric in 61% of the measurements.

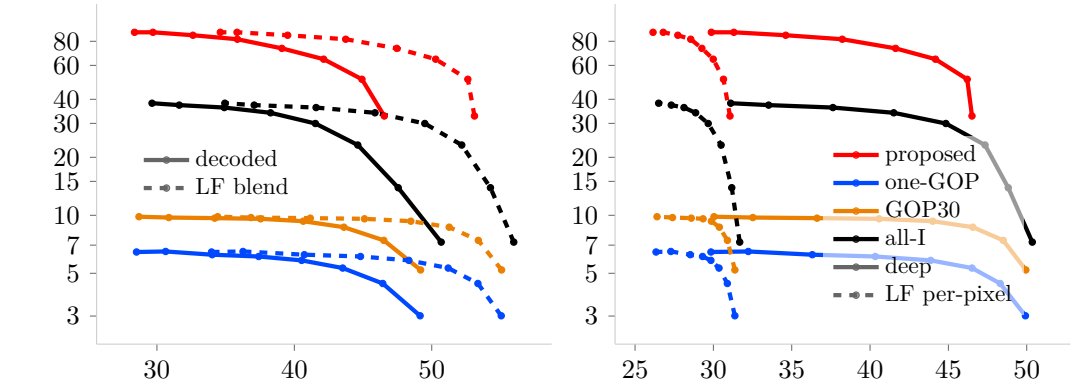
$$error = 100err_{visual} + err_{time} + err_{size} \tag{2.8}$$

The second metric is based on a threshold of 5% in visual quality error from the best result. This threshold was selected as a tolerable error that cannot be clearly distinguished by a human evaluator. The proposed scheme reached this limit in 58% of measurements.

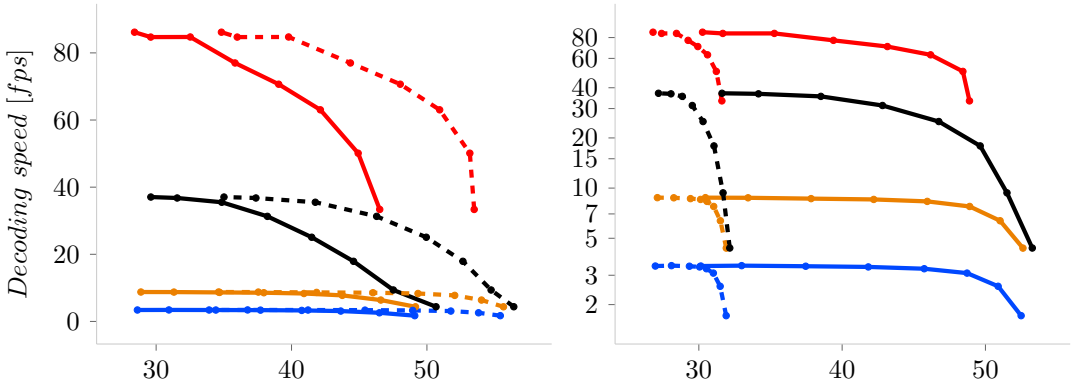
Tab. 2.3 shows the average decoding time, size, and quality metrics of the results. Given the significant speedup of decoding, the loss of visual quality is not critical. The disparity-related artifacts in the proposed scheme are mitigated when used in the light field rendering, and the scheme seems to perform better with larger and dense data due to higher inter-frame similarity.

		time	PSNR [dB], SSIM, VMAF												size
data	scheme	[ms]	decoded			deep			LF per-pixel			LF blend			[kB]
50	proposed	**15.84**	*37.41*	0.86	*69.86*	*39.07*	0.90	*81.46*	28.77	0.80	*35.69*	44.65	**0.96**	82.51	289.96
	one-GOP	195.25	38.78	0.87	*72.47*	40.92	0.92	83.71	29.24	0.81	40.15	45.47	**0.97**	83.47	**253.06**
	GOP30	136.65	38.91	0.88	73.22	41.10	**0.92**	84.29	**29.28**	**0.81**	**40.59**	45.72	0.97	83.92	258.55
	all-I	50.13	**39.85**	**0.88**	**76.81**	**41.89**	0.91	**87.31**	29.23	0.81	*38.16*	**46.42**	0.96	**86.20**	544.29
100	proposed	**16.16**	*37.37*	0.86	*69.79*	*40.40*	0.90	*84.17*	29.41	0.81	*37.81*	45.07	**0.96**	83.05	287.35
	one-GOP	349.93	38.96	0.88	73.29	42.53	0.91	86.29	29.92	0.83	43.21	45.91	0.96	84.09	**222.66**
	GOP30	160.00	39.16	**0.88**	74.37	42.80	**0.92**	87.09	**29.99**	**0.83**	**43.88**	46.39	0.96	85.42	228.76
	all-I	69.55	**39.81**	0.88	**76.69**	**43.53**	0.91	**89.43**	29.85	0.82	*40.52*	**46.77**	0.96	**86.37**	539.41
500	proposed	**17.02**	*37.38*	0.86	*69.76*	*41.16*	0.90	*84.97*	30.36	0.83	*49.55*	45.47	0.97	83.66	285.86
	one-GOP	1,420.28	39.36	0.88	75.13	43.32	0.91	87.75	30.98	0.85	57.47	46.47	0.96	85.79	**167.86**
	GOP30	144.05	39.71	**0.89**	**77.22**	43.81	**0.92**	89.36	**31.10**	**0.86**	**59.05**	**47.46**	**0.97**	**88.06**	180.03
	all-I	233.40	**39.78**	0.88	76.61	**44.05**	0.91	**89.95**	30.86	0.84	*53.51*	47.09	0.96	86.62	535.45

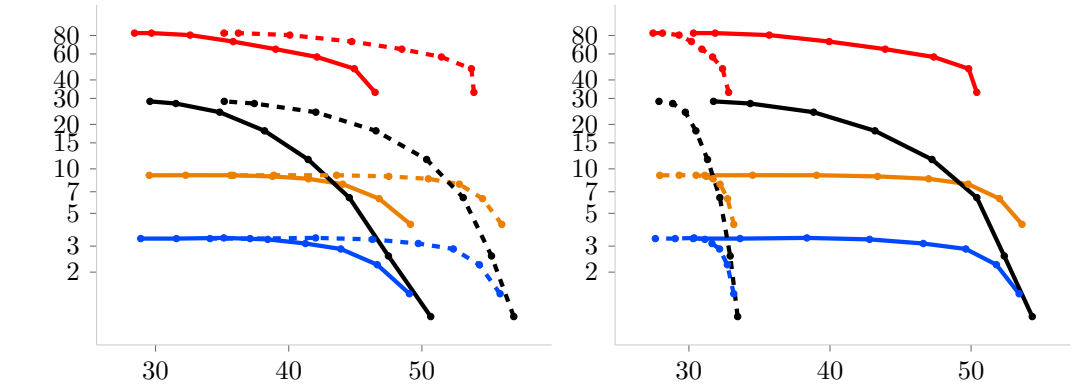
Table 2.3: The proposed scheme is the fastest, has the best quality/speed ratio, and the quality loss is not significant. The best absolute results are typed in **bold**. The lowest results of the weighted sum of errors, prioritizing visual quality, are underlined. The loss in visual quality compared to the best result greater than 5% is typed in *italic*.



(a) 50 frames from the dataset were compressed.



(b) 100 frames from the dataset were compressed.



(c) 500 frames from the dataset were compressed.

$PSNR$ [dB]

Figure 2.58: The video encoding schemes are compared in the charts. The measurements were performed on the c) full dense dataset, b) the sparser one where each fifth frame was taken and a) even sparser where each tenth frame is taken. Four comparisons were conducted: decoded frames as they were (left column solid lines), all frames blended together [158] (left dashed), deep interpolation [57, 127] (right solid) and per-pixel interpolation [9] (right dashed) which is most error prone due to the compression artifacts.

It even outperforms the other schemes in one case. Fig. 2.59 compares the visual quality at the same decoding speed. The amount of artifacts in the other methods is much more visible than in the proposal.

Figure 2.59: The figure shows the difference in visual quality between the proposal and other schemes while reaching the same decoding speed at *Pavilion* scene. The quality was adjusted so that the decoding times would match. The other methods had to use much more aggressive compression levels to reach the same decoding time as the proposal. The 10th frame was decoded from 100-frames dataset. The position of the frame was chosen to ensure a similar decoding time but also reasonable position within the light field (not at the very beginning or end). Frames further from I-frames in the other methods would not be able to compete with the proposal. Decoding multiple frames would put JPEG2000 at a disadvantage.

The following triplets of values are in the format of PSNR [dB], SSIM, VMAF. The best visual results were obtained with the *Bunny* scene, where the proposed methods yielded best results as 48.38, 0.9356, 96.59 and the worst as 32.23, 0.6655, 30.18. The worst results

were measured with *Class* scene with the best 45.52, 0.9872, 96.16 and the worst 25.60, 0.7213, 17.54. This shows that the complexity of the scene plays a role in the efficiency of the encoding, which is common for most existing compression methods. The *Bunny* scene contains only a simple object with constant background, whereas the *Class* scene contains a lot of small details in all parts of the scene.

The proposed scheme is not capable of reaching the same quality as the other schemes at the almost lossless level. The best results with *Bunny* scene are for one-GOP 53.04, 0.9966, 97.046, GOP30 53.17, 0.9971, 97.048 and all-I 55.40, 0.9981, 97.29. This reveals that the proposed method is not a good choice for archiving best-quality data. However, these best-quality settings are not suitable for real-time rendering, where a certain amount of hardly detectable information loss is to be expected.

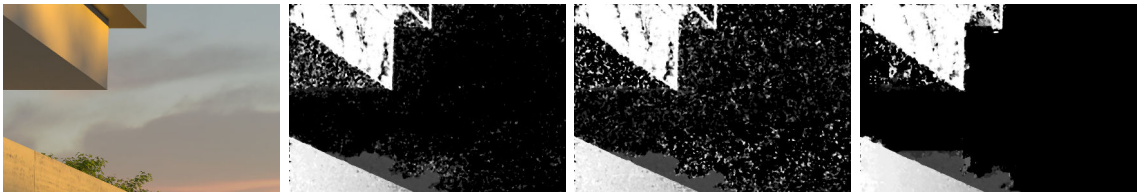
The average standard deviations over the quality range of the visual quality metrics are for the proposed method 2.41, 0.12, 5.35, one-GOP 2.37, 0.11, 3.94, GOP30 2.36, 0.10, 3.74 and all-I 2.67, 0.12, 5.23. These values show that the methods are all similarly suitable for all datasets without a significant divergence in any case.

The standard deviations of decoding times in milliseconds at a default $CRF = 28$ with *Bunny* scene are for the proposed method 0.779, one-GOP 648, GOP30 86.7 and all-I 14.4. This shows that the proposed method is the most stable and all packets have the same decoding time, excluding the first reference, which is almost 16× slower but is decoded only once during the initialization stage. The best and worst quality decoding speeds in frames per second are for the proposed method 28, 85, one-GOP 0.36, 13, GOP30 3, 14, and all-I 1, 31. The whole quality range of the proposed method can be considered suitable for real-time streaming.

The best and worst quality data sizes [MB] are for the proposed method 764, 2.4, one-GOP 486, 0.33, GOP30 506, 0.5, and all-I 1 461, 4.7 on *Sand* scene, which showed the highest variance of the sizes. The proposed method is not capable of reaching the same data reduction as one-GOP due to the impossibility of reusing information from decoded P-frames. Compared to all-I, which is second fastest after the proposed method, the possible data reduction is significantly higher.

The video encoding with one I-frame also helps to improve the quality of the light field rendering methods. The generated focus map in Fig. 2.60, from the per-pixel focusing method [9], shows how the inter-frame encoding aids the method. The focus map contains optimal displacement values between the images so that every pixel is focused at the correct depth. The map computation is based on color similarity between the displaced pixels of the views. The all-I-frame encoded frames are compressed separately and the compression artifacts that are different in each frame create bigger disparities even in the areas of the light field views that should be the same (the sky in the figure). When the P-frames are encoded using one common I-frame, their similarity in the static areas is untouched, and the compression even serves as a denoising for such areas. The disparity detection algorithms based on the color difference between pixels or their blocks then identify such areas even more accurately than when using the uncompressed original frames. This hypothesis is also confirmed in Tab. 2.3, where the all-I scheme does not always win in terms of visual quality during the interpolation process.

The discovered results are expected to be valid even for higher-dimensional light fields, which will be further investigated and proved in future work. The random access feature would be even more convenient in 2D light field grids where at least 4 views are necessary to decode. One row-wide offset between the two pairs would also require additional decoding.



(a) An image was used to compute a focus map.

(b) The focus map was estimated from the original.

(c) The focus map was estimated from the decompressed I-frame.

(d) The focus map was estimated from the decompressed P-frame

Figure 2.60: A part of a wall and the sky from *Pavilion* dataset is shown in a). The focus map produced by a light field rendering method of the original input in b) resembles a disparity or depth map, but light noise is visible in the static sky. All-I-frames in c) decoded and used in the method produce even more noise. The P-frames similarly used in d) produce the best focus map due to the inter-frame compression and reduction of color differences in static areas.

A test performed on 2D light field *Bunny* scene from Stanford dataset [267] indicates similar improvements as in the 1D measurement; see Tab. 2.4.

scheme	grid	speedup of proposed
all-I	17x17	2.066
	8x8	1.94
GOP30	17x17	8.64
	8x8	3.19

Table 2.4: The table shows how many times is the proposed scheme faster when decoding 4 views in the center of the grid (coordinates $(3,3),(3,4),(4,3),(4,4)$ for 8×8 and $(8,8),(8,9),(9,8),(9,9)$ for 17×17 grid) of a 2D light field scene, compared to all-I and GOP30 schemes encoded with the same default quality settings ($CRF = 28$).

The proposed scheme theoretically ensures the best memory usage since only the initial I-frame is kept in the memory. The memory usage on GPU was measured experimentally, decoding a frame at the end of the 100 frames light field. Tab. 2.5 shows that the proposal needs the least amount of memory. The all-I-frame decoding needs a bit more, probably due to the decoder allocating more memory in advance for further sequential decoding.

scheme	proposed	one-GOP	GOP30	all-I
used memory [MB]	597	1019	1049	800

Table 2.5: The table shows a comparison of the schemes at the same quality level $CRF = 35$ with 4K data.

2.8.6 Comparison to Existing Methods

An optimal light field data transfer to GPU was previously published, but it is not directly comparable with this proposal [154]. The idea of video encoding and the principles of the transfer scheme might be fused in a novel method in the future. The reordering of the views [103] before encoding could also bring about a certain improvement that is inferior to the proposed scheme. Previously published optimal subsampling of light field views [55] might be combined with the proposed video encoding. Compression and reconstruction networks [171, 215] or light field-friendly displacement intra prediction [165] use H.265 video encoding for light field data, so this proposal could be used to improve their performance. The proposed method is versatile and can be used as a part of other previously proposed methods.

Although deep central light field image warping [106] outperforms video encoders, the approach would be problematic with wide spacing of the views since occlusions in the scene might cause warping artifacts. This proposal supports newer and potentially faster decoding standards even with wide datasets. A deep generative model using a randomly initialized network can be used without learning for light field compression [129]. The proposed method does not depend on the training process.

GPU support is a crucial requirement as modern light field rendering methods and 3D simulations are computed mostly on GPU and the data upload to GPU is the performance bottleneck. This method focuses on achieving the best quality/decoding time ratio. Other approaches generally aim to achieve the best compression ratio [179] even with partial random access support [181].

An apparent candidate for this task - Multiview Video Coding (MVC) extension of H.265 that offers an optimal coding of animated multi-view sequences [72] is not fully supported by commonly used codecs and no plans for GPU HW support are expected. According to the current state of this extension, it cannot be expected to be present in more efficient GPU-accelerated formats such as (AV1) AOMedia Video 1. The proposed scheme does not depend on a specific format extension and uses the basic compression available in any video encoding format. The scheme can be used in any future format. Similar proposals [21, 271] to this method exploit the MVC extension, but it is not clear how efficient this approach would be in real-life scenarios on GPU. Compared to this method, the measured datasets were sparse, in a low resolution, the actual GPU decoding speed was not measured, and the decoded views were not used in any light field rendering method before comparison. Other specific approaches such as HEVC-HR [187] are most likely to be outperformed with the novel video formats. Although other custom light field compression methods exist [38], the proposed scheme is universal, scalable, independent on underlying video encoding method, and guaranteed to be efficient even on future GPUs.

The encoding speed or compression ratios are not relevant for this study. The real GPU-based decoding speed at the given quality plays the major role. Fig. 2.53 already showed how GPU-accelerated image compressions are outperformed by the video approach.

The untrained neural compression paper [129] reported decoding of 81 views of 432×624 resolution simultaneously, measured on GeForce RTX™ 2080 Ti, in 262 ms. The same scene *Danger de morts* of the pleno dataset [220] was measured using video compression with H.265 with one reference frame on GPU GeForce RTX™ 2070. 81 views can be decoded in 273 ms while achieving the same quality. The performance is the same. However, video decoders are improving and new coding standards are implemented, so the decoding speed might be even better soon. The natively HW-accelerated decoding proposed in the research

in this thesis can gain the same quality as a complex training-based method and might be a more robust solution. The authors in the mentioned paper measured *decoding time* and it is not clear if uploading of the data to GPU was also measured. The upload time was included in the measurement of the method proposed in the experiments, as it plays an important role in real-time streaming. They also mention that the network needs to be transmitted to the decoder for each light field, which brings an additional unmeasured delay. It is not clear how fast the decoding would be when only certain views are necessary.

Similar issues were found with neural video compression with diverse contexts [161] where decoding of 96 images takes 765 ms for 1920×1080 video on 2080 Ti GPU. Video compression is capable of reaching 634 ms for the same task according to the measurements conducted. The same experiment with 96 images was conducted with the published code of learning for video compression paper [293]. The same issues prevail, and the decoding took minutes. The decoding of a frame took approximately 10 s.

A dictionary learning approach paper [184] reports reaching 40 fps for a 2048×1088 light field data, measured on NVIDIA® Titan Xp. The proposed decoding scheme reaches up to 80 fps with almost 2× larger images. The mentioned method also supports random access. The disparity-aware learning-based model [241] outperforms other methods but does not provide a significant gain in terms of decoding speed. This method reaches decoding times for low-resolution Lytro data in tens of seconds, which is not comparable to fast GPU decoders. The methods mentioned above were designed for Lytro camera data, while the proposal aims at data with a wider spacing between the views. Compared to Lytro data, where the maximal disparity reaches tens of pixels, the proposed method can process disparities of hundreds of pixels.

Dual discriminator generative adversarial network compression [23] reaches certain quality gain, but the decoding takes tens of seconds due to additional synthesis of the missing frames from VVC encoded reference frames. Bi-level view compensation compression [116] is almost 1000× slower than H.265 at decoding. Authors of adversarial network-based view synthesis paper [125] also admit that the method cannot be used in real-time. Various JPEG extensions to 3D and 4D for multiview data were also beaten by video compression methods when used with light fields [1, 2].

Chapter 3
Hypothesis & Experimental Proof

This chapter defines the core scientific contribution of this thesis. It consists of a scientific hypothesis which is defined to address the main issues related to light field rendering. These issues are currently not efficiently solved by existing state-of-the-art methods. The hypothesis is followed by sections describing the experiments that were implemented as its proof.

3.1 Hypothesis

The main scientific contribution is related to the topic of light fields. The scientific goal of the thesis is to experimentally prove the proposed Hypothesis and demonstrate progress beyond the state of the art.

Hypothesis *A novel method of light field rendering can be designed and implemented that produces all-focused renders directly from input views without exploiting additional depth information, having better visual quality, being computationally more efficient, and having less memory requirements compared to the state of the art.*

The proposed method was implemented, and experiments were carried out to measure its performance in visual quality, time, and memory requirements. The results obtained prove the proposed Hypothesis. The implementation is GPU-accelerated to exploit the contemporary hardware support and enable real-time usage of light fields in industry.

This thesis addresses the main issues that were not efficiently solved in previous state-of-the-art light field rendering methods. The results presented lead to a complete solution that solves all the light field rendering issues at the same time. Previous proposals usually focused on selected aspects without taking into account the whole process of light field rendering. This makes the usage of light fields in real-life scenarios difficult without the proposals in this thesis.

Light field assets are currently not widely used even though they achieve potentially better rendering time performance than state-of-the-art rasterization or ray-tracing methods. The light field rendering time is near constant and does not depend on the content of the scene. Even complex materials and scenes can be quickly rendered using this image-based approach.

The aim of the original research described in this chapter is to create a solid ground for light field assets that can be widely used in practice. The desired result of the proposed method is a novel synthetic view generated from the input set of views representing a discrete light field approximation. The proposed method is designed for widely spaced

discrete light field data. Narrowly spaced data, for example, taken by plenoptic cameras, are useful for refocusing of the final photo at different distances. Widely spaced data allow for 3D movement in the scene [283]. Such assets are meant to be used in 3D applications or cinematography for rendering of arbitrary 3D scenes in real time. The following goals needed for convenient light field rendering in industry are addressed in this chapter:

- Limited memory requirements.
- Real-time playback with quality results that allow for efficient GPU implementation.
- All-focused view without the need for pre-processing of input images or additional information about the scene.

The following sections are based on original papers that were published as incremental results of the research carried out. These are the core papers describing the novel method proposed in this thesis.

3.2 Real-time Per-Pixel Focusing

This section is based on an original method of the author [9]. The basic method described here addresses the claims of Hypothesis that the proposed method is computationally more efficient than existing methods and does not require additional depth information about the input scene.

This section describes a real-time light field rendering method that can be used with contemporary consumer hardware. The inputs of this method are a discrete light field approximation, as a grid of images, and a position of the virtual camera. The output is a novel view that is focused in all parts of the scene. Various open-source applications were implemented during this research and an interactive web demo[1] is available. The main principle, allowing for the real-time usage, is the design of a highly parallel optimal algorithm for GPU.

First, a focus map with the optimal focus distance is computed for each pixel. Then, each pixel of the novel view is computed as a combination of pixels from the input images, so that the contributing pixel colors are sampled from the same spot in the scene. The focus map contains the mutual displacement of the images to achieve the all-focused sampling and simulates the scene surface. The focus map can resemble a depth or disparity map, but it does not contain the real depth of the scene. The scene structure does not have to be visible on the focus map; see Fig. 3.1.

Figure 3.1: The first image is a depth map. The second one is a focus from the proposal. It differs from the depth, but leads to artifacts-free novel view in the third image.

[1] The demo is available as a part of presentation website for one of the original papers [10] here: fit.vutbr.cz/∼ichlubna/lf

Compared to the disparity map, the focus map contains the offsets for all light field images and is not computed only for stereo pairs [102]. The values in the focus map are calculated for the new synthetic view and are not explicitly aligned with any of the input images. Color similarity is the main metric used during the focus map estimation as the scene geometry correspondence in the views is not available. The main goal is to minimize the final rendering error. Therefore, focus maps are tailored to be used in the light field image-based rendering but not for 3D surface or point-cloud reconstruction. The scene can be viewed from an arbitrary point of view within the bounds defined by the capturing grid. Fig. 3.2 shows the principle of focus map usage in light field rendering.

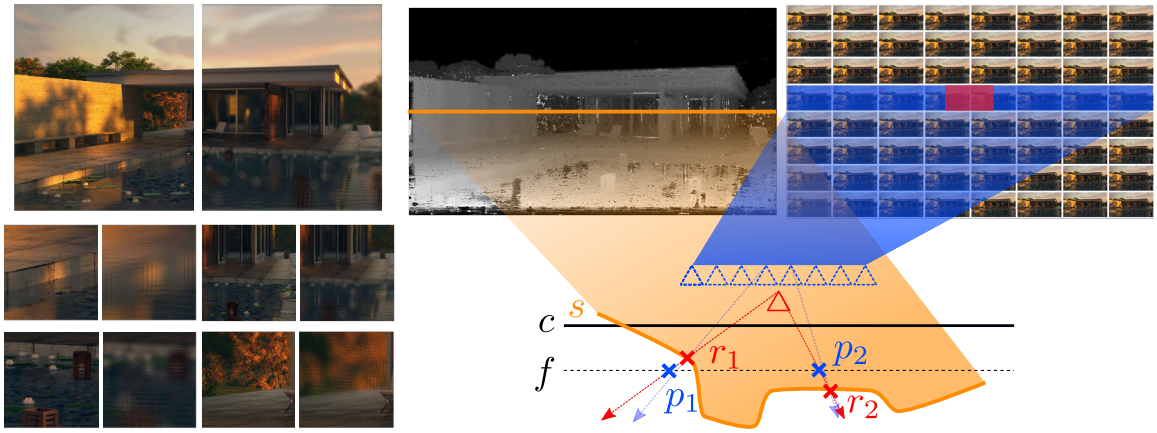


Figure 3.2: Left: Comparison of fully focused light field image as a result of the proposed method with light field focused at a single distance. Right: From the input set of images taken by camera grid, a new synthetic view of the scene is generated having every location in the scene focused as if captured by a pinhole camera. focus distance values for each pixel are estimated and stored in a focus map which resembles disparity or depth map of the scene. This map is used to achieve a correct focusing of each pixel in the novel view synthesis.

Compared to previous methods [18, 41, 160, 239, 247, 266], the proposed algorithm is designed to optimally utilize GPU for real-time rendering scenarios and also outperforms the methods in the resulting visual quality. It works well with sparse, dense, and high-resolution light field datasets with no excessive hardware requirements. As proved in the experiments, the method also reaches better quality than methods in the same category and even reaches similar quality levels as deep learning approaches. Deep learning usually depends on the long and expensive training process, and the rendering, using the pre-trained model, is not reaching the times and memory requirements suitable for real-time usage with high-resolution data.

The proposed method is lightweight in all aspects. No additional depth information is needed as the input, which also saves memory and allows for the usage of the input images without preprocessing. One of the notable advantages of the proposed approach is its independence from the scene content. The computation time is constant and depends only on the desired resolution of the resulting view and quality settings. This method can be used, for example, to render scenes behind windows or in a far distance in computer games where the player's movement is limited by the environment. It can also be used to provide a real-time preview of light field video, for example, on a film set, when light field capturing technology is used.

3.2.1 Per-Pixel Scanning

The key part of the method is the generation of the focus map. It uses principles similar to stereo disparity estimation methods [112] but compares colors from multiple images of the input light field grid. For each pixel, the method iterates over a focus distance range, interpolates the sampled colors from the input images at the given focus distance, computes the variance of the colors used, and stores the focus distance with the lowest color variance. The weighted shift-sum algorithm [13] is used for color interpolation, where the weights are computed by a linear function of the distance of the sampled image from the position of the virtual novel view in the grid. The focus map then contains for each pixel its lowest-variance focus distance. The variance is computed based on the Chebyshev distance between the pixel values. Generally, the choice of the color distance metric for a given task is problematic [227]. Chebyshev was experimentally proven to be the most suitable metric in this case. The whole process is described in Alg. 1.

```
Data: Position of virtual camera, grid of images, focus range
Result: Focus map, novel image
foreach coordinate ∈ image do
    variances = emptyArray();
    colors = emptyArray();
    foreach focusDistance ∈ range do
        newPixel = shiftSum(camera, focusDistance, coordinate, grid);
        variances[focusDistance] = newPixel → variance;
        colors[focusDistance] = newPixel → color;
    optimalDistance = indexOfMinimum(variances);
    map[coordinate] = optimalDistance;
    image[coordinate] = colors[optimalDistance];
```

Algorithm 1: Function shiftSum() uses the shift-sum algorithm and returns the final color and variance of the colors contributing in the summation. This algorithm is generalized, returning also the focused pixel color. However, image synthesis is separated in the reference implementation.

In terms of 3D cost volume [93], the focus distances are the hypothesis planes, the spatial resolution of the focus map is the spatial resolution of the cost volume, and the cost function is the variance of the colors in the given cell. When the focus map is ready, the shift-sum algorithm is used again to synthesize the final image, where each pixel is interpolated at the previously computed optimal focus distance. The focus map computation and rendering of the novel view are separated in two stages so that optimizations such as focus map subsampling can be used. The example of both the final image and the focus map is shown in Fig. 3.3.

The 3D effect is simulated by the weights assigned to the input images and the novel view camera coordinates that are used in the shift-sum algorithm to define the relative displacement of the input images. The color vector of the output pixel $\mathbf{p}_o$ can be obtained by iterating over n input images. The function p_i in Eq. 3.1 returns the pixel of the ith image at the given coordinates. Each input color is weighted by w_i. The output view coordinates $\mathbf{c}$ are used to shift the sampled images by the offset $\mathbf{o}_i$ between the images in the grid. The amount of shifting is adjusted by the desired focus distance f.

(a) A view is rendered from light field. (b) A focus map is used in the process.

Figure 3.3: Focused result using the *Pavilion* dataset with a correspondent focus map are shown side by side. The focus map contains estimated focus distance for each pixel of the final image. The map resembles depth or disparity map for the given synthetic view because the focus distances depend on the distance of the geometry one each pixel from the camera.

$$\mathbf{p}_o = \frac{\sum_{i=1}^{n} p_i(\mathbf{c} + \mathbf{o}_i \cdot f) \cdot w_i}{\sum_{i=1}^{n} w_i} \tag{3.1}$$

An iteration of the used shift-sum algorithm is depicted in Fig. 3.4.

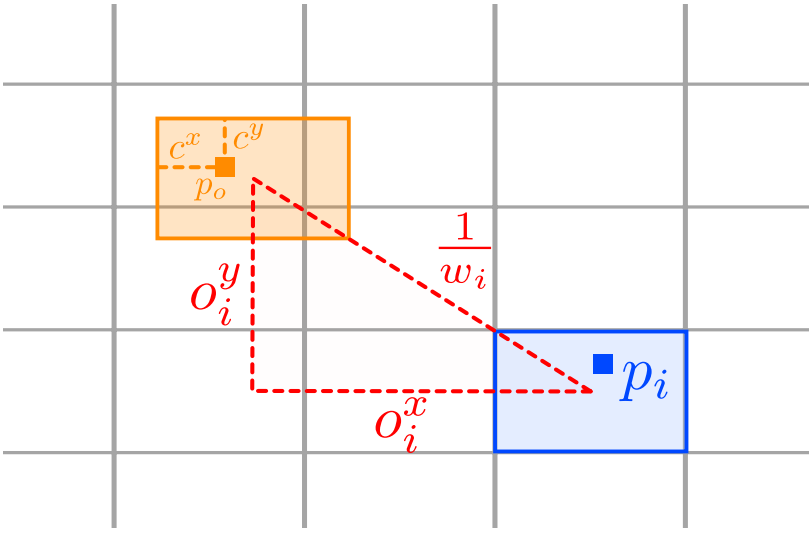


Figure 3.4: The figure shows one iteration of the shift-sum based image synthesis where a pixel from *ith* image (blue rectangle) is taken into the summation of the output pixel $\mathbf{p}_o$. The orange rectangle depicts the new synthetic image. The red lines show the offsets relative to the currently sampled image and the distance between the two images. The currently sampled pixel's weight w_i depends on the distance between the two images. The x and y superscript denotes the first and the second element of the vector variable.

3.3 Improvements of the Focusing Algorithm

Additional improvements in quality and performance, optimal scheme for light field video playback, and automatic detection of important parameters of the original method were further proposed and published [8]. Hypothesis claims that the proposed method achieves better visual quality than the state of the art. The improvements described here mitigate the artifacts that can occur when the focus map contains inaccurately estimated focus values for some pixels.

Objects of one-pixel size are usually not present in ordinary scenes, and such pixels can be assumed to be focused at the same distance as their neighbors. The first straightforward enhancement of the focus map is filtering. Median filtering is used to improve the quality of the focus map, as shown in Fig. 3.5. A GPU-optimized min-max median algorithm [225] is an efficient solution.

Figure 3.5: The noise in focus map resembles salt-and-pepper noise. Median filtering can be used to get rid of incorrectly focused pixels.

3.3.1 Variance Course Analysis

False focus distance detection in the shift-sum algorithm is the source of visual quality degradation. Large segments filled with a single color, repetitive patterns, or coincidentally similar areas around one pixel can be problematic. An analysis of the variance course for one pixel in the problematic area of the image (thin edge) can be seen in Fig. 3.6. The global minimum does not correspond to the correct focus value. However, the desired detail causes a peak in the variance chart. This steep change in the variance value, which is close to the minimal value, can indicate such details. After the variances are computed, a fast shuffle operation for interthread communication on GPU can be used to test each triplet

of neighboring variances, detecting the difference that is above a certain threshold. This scheme expects each GPU thread to evaluate one focus distance. Prioritizing the focus distance, which is low enough and at the bottom of a saddle, reveals some of the previously distorted details in the image. The method is described in Alg. 2.

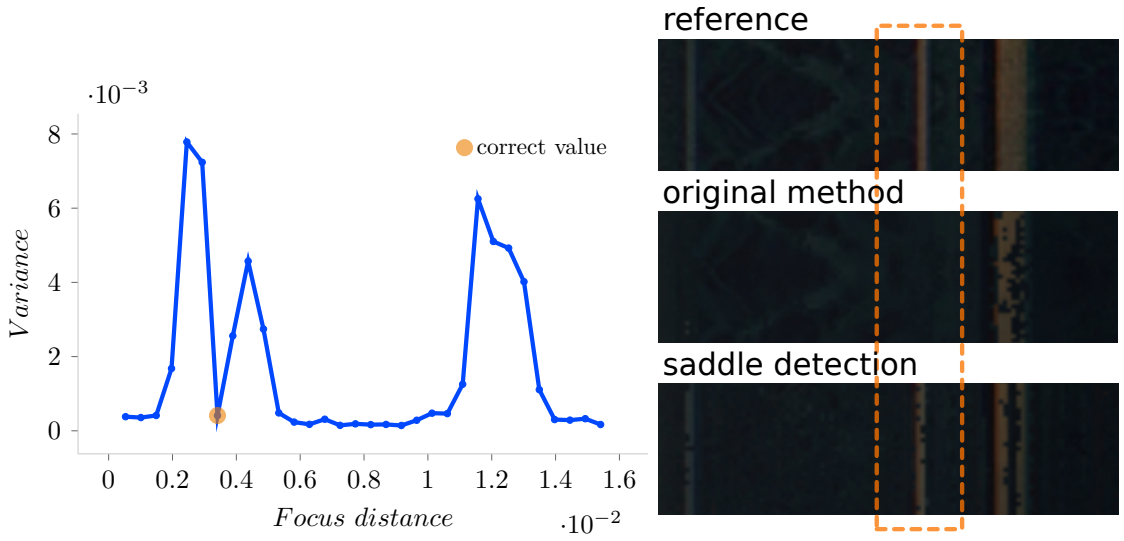


Figure 3.6: The chart shows the course of the variance coming from the shift-sum for one pixel. The global minimum value does not correspond to the best focus distance in this case. Orange node indicates the correct distance. The steep saddle is a thin detail, the frame of the window highlighted in the picture.

The saddle detection reveals problematic details, but it can lead to a false positive and cause artifacts in the image, especially near low-frequency areas. Many scenes contain various kinds of background, such as sky, walls, etc. Such backgrounds are bound to be in focus at the beginning or end of the focus range. Sampled distances at the boundaries of the range can converge to very low variance values. If the values are small enough and their difference is below a certain threshold, it can be assumed that the tested pixel is in focus at the border of the range. Alg. 3 describes this phase, and an example is shown in Fig. 3.7.

3.3.2 Pixel Sampling

Instead of sampling one pixel per scan from the transformed coordinates, sampling multiple pixels at a certain distance can lead to better results. It is based on the assumption that pixels that are adjacent or very close neighbors belong to the same object in the scene. This block matching resembles matching cost computation and aggregation used in stereo disparity estimation methods [230]. However, increasing the number of texture-reading operations slows down the computation. The pixels can be sampled using various patterns. Fig. 3.8 shows the best patterns chosen according to the performance results.

Data: Array of $variances$ associated with focus distances, $sensitivity$
Result: Edited array of variances $editedVariances$
$editedVariances = \text{emptyArray}()$;
$editedVariances \xleftarrow{\text{insert}} \text{firstElement}(variances)$;
$max = \text{maximalElement}(variances)$;
foreach *three consecutive variances* $var_{previous,current,next} \in variances$ **do**
    $var_{new} = 0$;
    if $var_{current} < var_{prev}$ **and** $var_{current} < var_{next}$ **then**
        $norm = \frac{var_{current}}{max}$;
        $prev = \left| norm - \frac{var_{prev}}{max} \right|$;
        $next = \left| norm - \frac{var_{next}}{max} \right|$;
        $saddle = prev + next$;
        $var_{new} = var_{current} - saddle \cdot sensitivity$;
    else
        $var_{new} = var_{current}$;
    $editedVariances \xleftarrow{\text{insert}} var_{new}$;
$editedVariances \xleftarrow{\text{insert}} \text{lastElement}(variances)$;

Algorithm 2: Saddle detection algorithm that prioritizes local minimum over the global one in the array of color variances during the focus map generation phase. The bigger the difference between the minimum and its neighbors, the more the minimum shifts to outperform the global one.

Data: Edited array of variances from saddle detection phase $editedVariances$, number of border elements to check $checkNum$, detection $threshold$, $tolerance$ marking the same variance level
Result: Edited array of variances $editedVariances$ with new minimums
$index = 0$;
if $\text{firstElement}(editedVariances) > \text{lastElement}(editedVariances)$ **then**
    $index = \text{sizeOf}(editedVariances) - checkNum$;
$same = false$;
if $editedVariances[index] < threshold$ **then**
    foreach $\{i \in \mathbb{Z} \mid 0 \leq i < checkNum - 1\}$ **do**
        $j = index + i$;
        $diff = |editedVariances[j] - editedVariances[j+1]|$;
        if $diff > tolerance$ **then**
            $same = false$;
            break;
    if $same$ **then**
        $editedVariances[index] = -1$;

Algorithm 3: The variances are adjusted by prioritizing the values at the borders of the course which are low enough and surrounded by same or similar values. Such cases occur, for example, when a distant background is present that stays in focus for a number of distances at the end of the range.

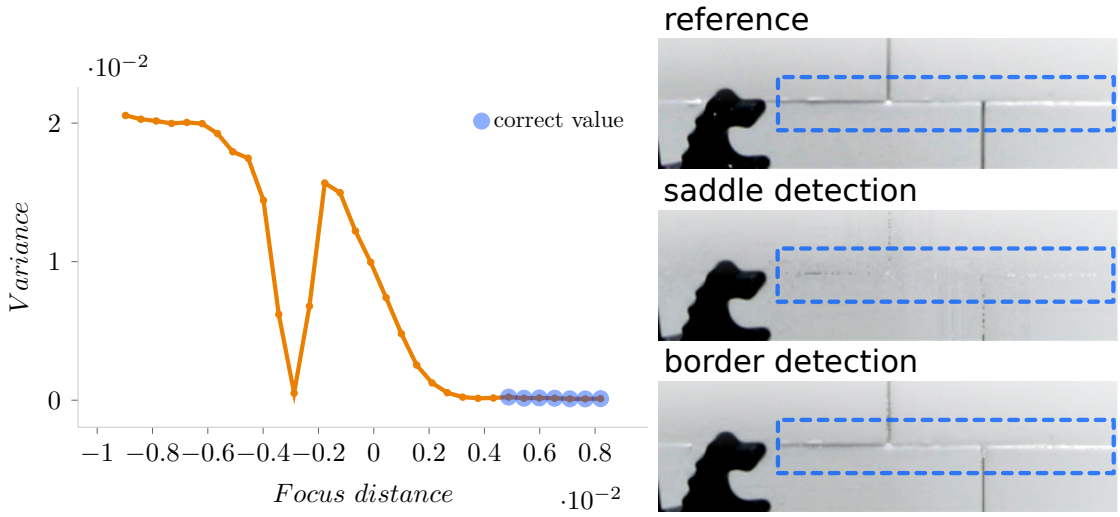


Figure 3.7: The chart is showing the course of the variance coming from the shift-sum for one pixel. The saddle point in this case causes incorrect focus as shown on the picture. The blue nodes are correct.

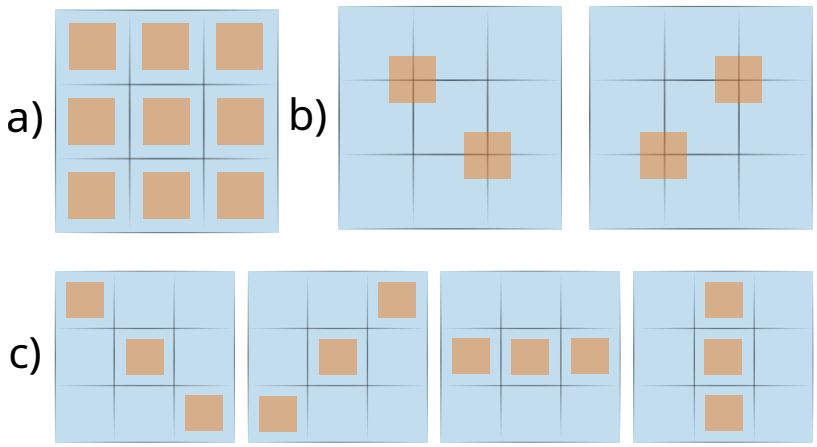


Figure 3.8: The figure depicts three tested texture sampling patterns: a) is a full 3x3 window, b) exploits the texture interpolation units in GPU, getting colors from coordinates between four pixels (similar to Kawase blur [139]), c) alternates over modulo, using four three-read patterns. Note that these patterns were also used in 2 px radius version.

3.3.3 Focus Range Detection

Predefined focus range can be inconvenient for dynamic scenes with a variable depth range. The focus range might also not always be known for the given light field. However, the range is an important parameter for the scan during focus map generation. Automatic fast detection of the start and the end of the range is necessary. An extra wide range is defined to ensure coverage of all standard datasets. Then this range shrinks into a tighter one. The image is divided into 16×16 pixel blocks. Variance is computed with the shift-sum algorithm in the same way as in the focus map generation. The variances are averaged through the entire block at each distance. The minimal average variance is then chosen as optimal. The minimum and maximum of these minimal averages from all blocks are selected as the final focus range.

Only four nearest central images are used in this computation. Using images that are far from each other, such as corner images of the whole grid, yields inaccurate results because of the reduced similarity between them. The local minimum variance of each 16×16 block is stored in an array, and then the global minimum and maximum are found; see Alg. 4. The array of local minimum variances is statistically filtered using the standard score method to remove outliers.

```
Data: Four neighbouring input images, scanning distance, size of one scanning step
Result: narrowed focus range
coef = 1/distance²;
range = 2DVector(distance, −distance);
foreach 16×16 block ∈ result do
    bestFocus = 0;
    minVariance = maximalValue();
    foreach fi from −distance to distance with step do
        f = coef · fi³;
        variance = 0;
        foreach pixel ∈ block do
            colors = emptyArrayOfColors();
            avgCol = emptyColor();
            foreach {i ∈ ℤ | 0 ≤ i < 4} do
                colors[i] = sample(images[i], pixel_xy, f);
                avgCol = avgCol + colors[i];
            avgCol = avgCol/4;
            foreach {i ∈ ℤ | 0 ≤ i < 4} do
                variance = variance + distance(avgCol, colors[i]);
        if variance < minVariance then
            minVariance = variance;
            bestFocus = f;
    range = minMax(bestFocus, range_max);
    range_min = min(bestFocus, range_min);
```

Algorithm 4: Automatic range detection iterates over a wide focus range. The new narrow range is constructed using focus distances with minimal variance of each 16×16 block of pixels.

Experiments showed that non-linear focusing scan with a constant step is more robust for this method. The desired focus range is always defined around the zero focus distance.

Therefore, this area is searched more densely according to Eq. 3.2, 3.3, and 3.4. This is achieved by transformation of the linear range, using a third degree polynomial, with only the highest degree coefficient set as non-zero as follows:

$$f(x) = ax^3. \tag{3.2}$$

The transformation of the linear value into exponential space is calculated as:

$$f_e = Af_l^3. \tag{3.3}$$

The coefficient A is calculated using the upper limit of the range as:

$$A = \frac{1}{f_{max}^2}, \tag{3.4}$$

where f_l is the focus distance in the linear range, f_e is the focus distance in the exponential range, and f_{max} is the upper limit of the range.

3.3.4 Depth of Field

The focus map can be used to generate post-processing effects, such as the depth of field. The defocused area is distorted by sharp ghosting artifacts in many light field applications [158]. The artifacts can be mitigated with a blur filter that smoothens the defocused parts. In the reference implementation, a two-pass separable linear blur was used, where the amount of blur linearly decreases as the focus distance is closer to the chosen target distance; see Alg. 5.

Data: All-focused light field rendered *image*; focus *map*, DOF *range*, DOF *distance*, size of the blur kernel *kernelSize*, coordinate-space size of the half of the pixel *halfPixel*
Result: Rendered *image* with depth of field effect

```
focus = round(distance · totalLevels);
for pass ∈ {x, y} do
    foreach pixel ∈ image do
        coord = pixel → coord;
        dist = |focus − sample(map, coord)|;
        kernelSize = round(dist · range);
        w = kernelSize + 1;                                   // weight
        color = sample(image, coord) · w;
        foreach {i ∈ ℤ | 0 ≤ i < kernelSize} do
            o = 2DVector(0, 0);                               // offset
            o[pass] = (3 + 4 · i) · halfPixel;
            w = kernelSize − i;
            color = color + sample(image, coord + o) · w;
            color = color + sample(image, coord − o) · w;
        color = color / (kernelSize + 1)²;
        outImage[coord] = color;
    switchImages(image, outImage);
```

Algorithm 5: Simulation of the depth-of-field effect on the rendered light field image uses the computed focus map. Half-pixel offsets are used to sample pixels for two-pass separable linear blur, exploiting the interpolation of texturing units in GPU.

The results in Fig. 3.9 show that this method can be extended to support multiple focus distances. The proposed improvements to the focus map reduce the number of artifacts in this effect; see Fig. 3.10.

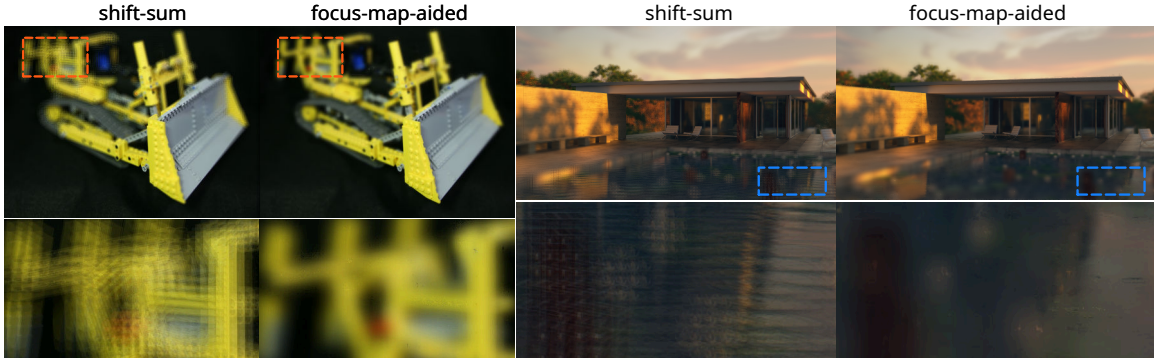


Figure 3.9: Focus map was used to simulate the depth of field effect, applying box blur on the defocused parts of the scene. A comparison with the original shift-sum algorithm is shown in the figure. Box-like artifacts are visible without the proposed improvements.

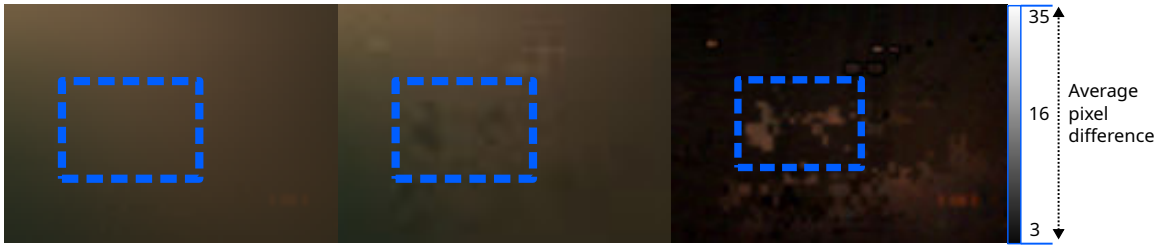


Figure 3.10: The left image shows a part of the depth-of-field effect in *Pavilion* scene using the focus map with the proposed improvements. The effect in the right image uses the focus map generated without the proposed improvements and contains more visible artifacts. The rightmost image is a difference between the top images.

3.3.5 GPU Utilization

The method is designed to exploit massive parallelism available on GPU architectures, which is important for the desired real-time usage. The focus map generation is performed in a GPGPU compute shader or kernel. Each warp (32 threads on NVIDIA cards) is assigned to one pixel. Each workgroup consists of 8 neighboring pixels. This scheme offers a good GPU occupancy and memory access coherency, allowing an in-warp data transfer between the threads, which is much faster than using the global or local memory. Each thread computes one focus distance (or more when denser search is required), using the weighted shift-sum and the Welford's variance algorithm [281]; see Alg. 6.

This way of variance computation improves the GPU occupancy by reducing the necessary number of registers. In the end, the minimal variance value within a warp is found using parallel reduction with ballot operation. In the fragment shader, a surface representing the light field is rendered. The focus map and the input images are stored as textures. Therefore, the missing pixels can be interpolated in texturing units if the resolutions of the result and the focus map differ. Fig. 3.11 describes the work distribution on GPU.

Data: Stream of *input* pixel values
Result: Estimated variance m_2
$n = 0$;
$mean = 0$;
$m_2 = 0$;
foreach $pixel \in input$ **do**
    $n = n + 1$;
    $delta = pixel - mean$;
    $distance = \text{pixelDistance}(pixel, mean)$;
    $mean = mean + delta \,/\, n$;
    $m_2 = m_2 + distance \cdot \text{pixelDistance}(pixel, mean)$;
$m_2 = m_2 \,/\, (n - 1)$;

Algorithm 6: Welford's method for computing online variance in one pass, adjusted to pixel values (RGB colors in reference implementation) comparison purpose. This algorithm is used in the shift-sum, analyzing new color values coming into the summation.

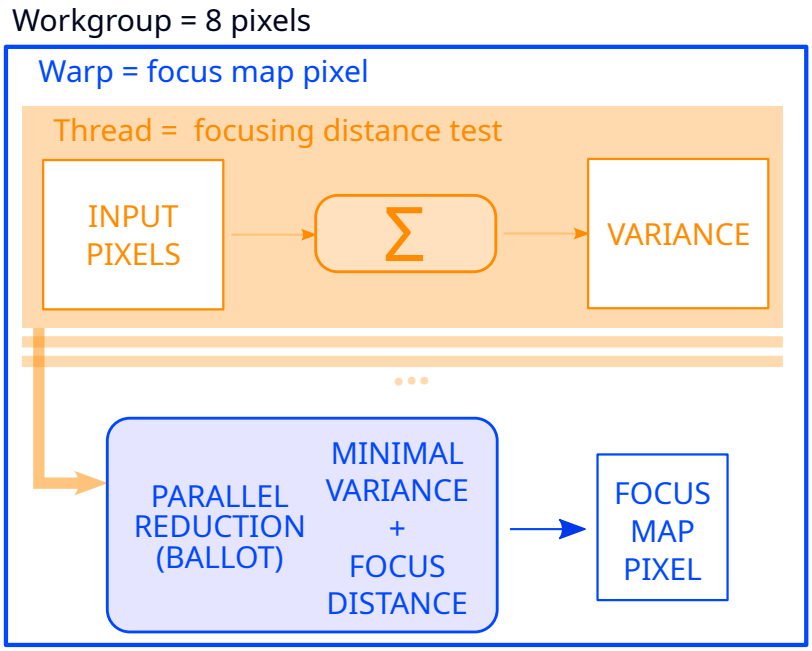


Figure 3.11: Work distribution on the GPU for focus map generation. The compute shader analyses the input images, going through the focus range and saving the focus distance with a minimal variance in the focus map. Because the workload is divided into warp-sized elements, no global or local synchronization, that would slow down the computation, is needed.

Video Streaming

Classic datasets [267] contain frames in the 8×8 grids. Current video standards expect a resolution of at least 1920×1080 (FulllHD), slowly rising to 3840×2160 (4K). This leads to almost 10 GB/s when streaming 25 fps FullHD 8-bit RGB video. Such a transfer from RAM memory via PCI-Express into GPU memory is hardly achievable in real time.

Video compression was shown to be an optimal solution for light field data [1]. It was discovered that compression artifacts are mitigated by light field rendering. The same fact was proven for downscaling with H.265 compression; see Fig. 3.12. PSNR difference is higher when comparing the raw data frame-by-frame than when comparing synthetic views produced by the light field rendering method. SSIM seems to be a bit worse in the light field case, but the difference is very small.

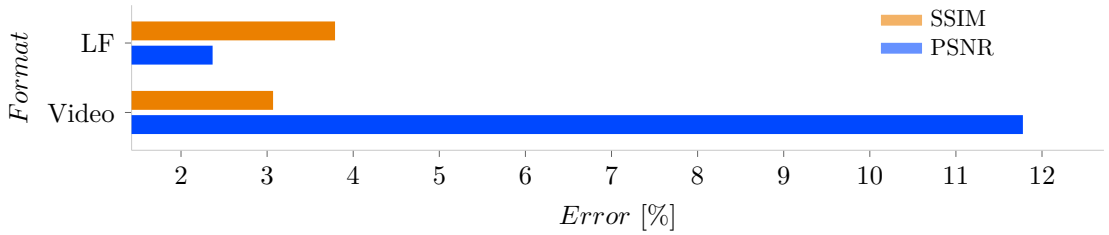


Figure 3.12: An average error, over all compression levels defined by H.265 CRF parameter, of PSNR and SSIM comparison of downscaled (HD) and original (FullHD) light field dataset. The downscaled dataset was compared to the original frame-by-frame as a video and as a light field, comparing the new synthetic views, produced using the original and downscaled input.

Video packets can be uploaded asynchronously, decompressed on GPU, and converted[2] to textures. The scheme in Fig. 3.13 describes the approach used in the experimental implementation.

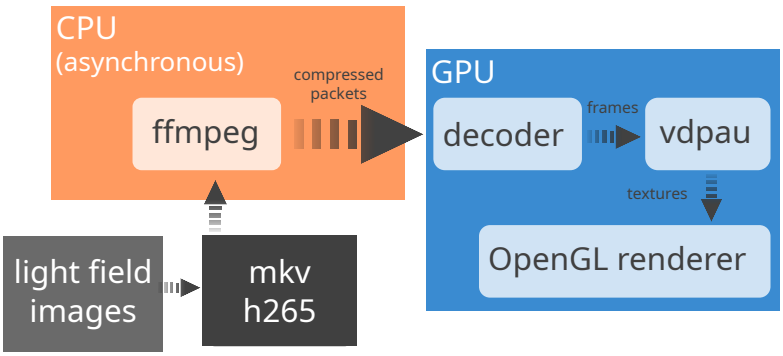
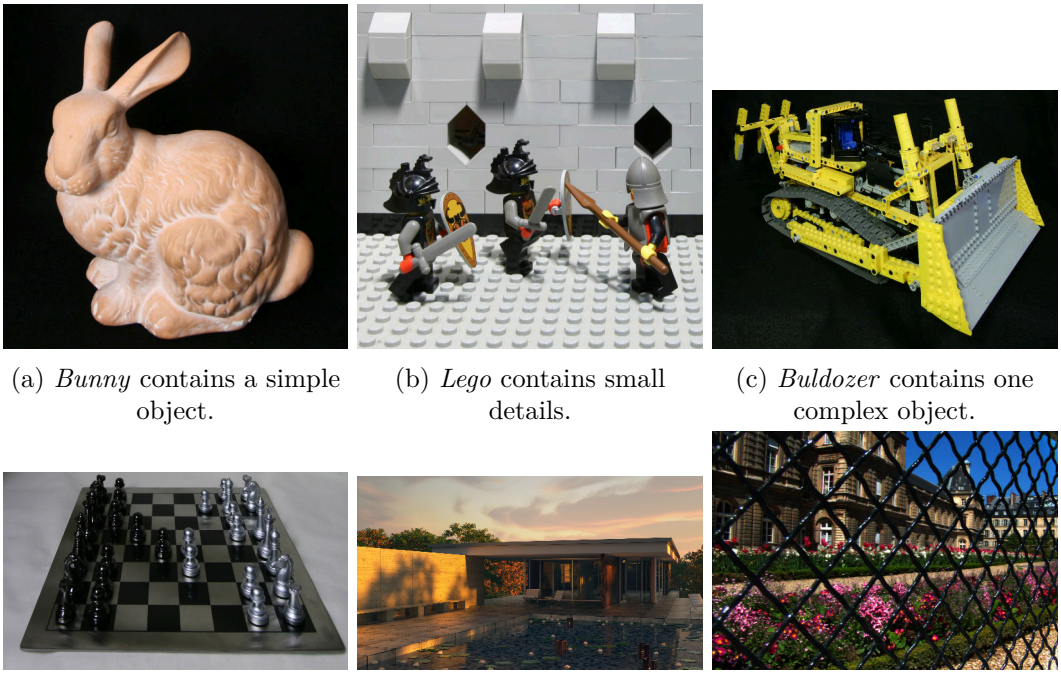


Figure 3.13: Light field images are compressed in a video stream which is transferred asynchronously to the GPU. Video packets are decoded on GPU, using vdpau, and converted into OpenGL textures. Textures can be then used for rendering, using classic rendering pipeline and compute shaders.

[2] GL_NV_vdpau_interop extension for OpenGL can be used.

3.4 Experimental Evaluation with Standard Datasets

The proposed method was evaluated using an experimental implementation. Datasets used in the experiments captured with camera array come from Stanford light field archives [267], light field captured by the plenoptic camera Lytro Illum belongs to EPFL light field dataset [220], and synthetic dataset was rendered on *Barcelona Pavilion* scene, which is available at the Blender demo files page; see Fig. 3.14. Only one Lytro dataset was used because the distance between Lytro views is very small and cannot be used to create an appreciable 3D effect. In all experiments, a ground truth center view from the original dataset was chosen as a reference and it was compared using SSIM and PSNR metrics to a new synthetic view rendered by the proposed method. All experiments were executed on a machine equipped with NVIDIA® GeForce RTX™ 2070 GPU and Intel® Core™ i5-8500 CPU 3.00 GHz CPU, running Arch Linux.



(a) *Bunny* contains a simple object.

(b) *Lego* contains small details.

(c) *Buldozer* contains one complex object.

(d) *Chess* contains reflective materials.

(e) *Pavilion* scene is a complex outdoor scene.

(f) *Lytro* is an outdoor scene taken by a plenoptic camera.

Figure 3.14: The images are reference views from each dataset used in the experiments.

3.4.1 Color Distance Metric

The variance computation phase in the proposed algorithm requires a pixel color value distance metric to decide how much two pixels differ in terms of color similarity. The right choice of the metric depends on various aspects, such as the expected color range, the type of images, or a final use case. Various color distance metrics were compared to find out which one would yield the best visual quality results for light field datasets; see Fig. 3.15. The quality differences were not significant, but the computational complexity of the metrics differed and could negatively affect performance. Chebyshew metric was chosen for further experiments because of high-quality results and computational simplicity.

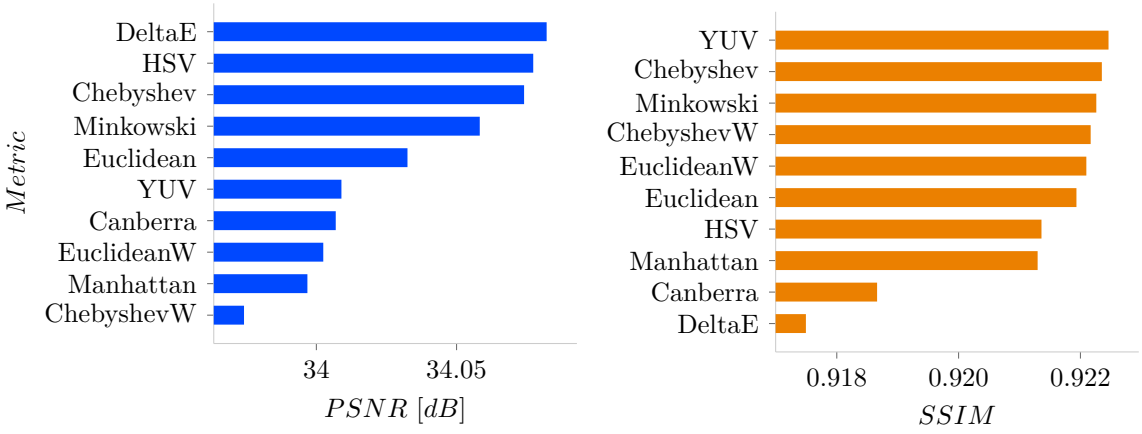


Figure 3.15: A comparison of the RGB color distance metrics for the pixel similarity test during the variance computation phase. The W suffix in the metric name stands for weighted metrics. The average results of all the datasets tested are presented.

3.4.2 Base Quality

For each dataset, the best focus range and search step were manually found and the resulting images were compared to the reference. The final visual quality is evaluated in Fig. 3.16. The images are focused in all parts, but interpolation artifacts are visible around thin edges or near similarly colored areas.

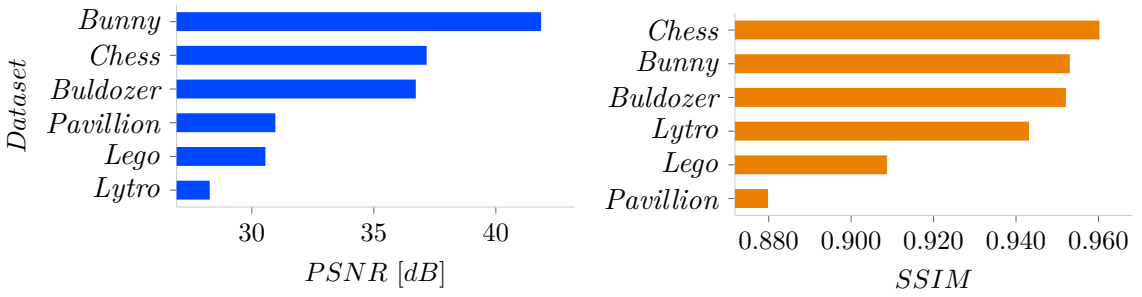
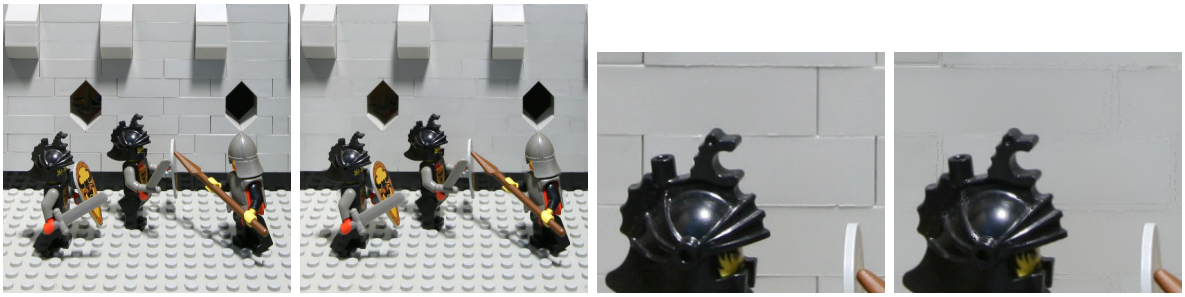


Figure 3.16: Best results are presented of rendering a new view from each dataset compared to the ground truth.

A detailed look at the interpolation artifacts is captured in Fig. 3.17. *Bunny* dataset contains only diffuse material and is clearly separated from the black background. Therefore, the reconstruction had minimum artifacts. Although the *Chess* dataset contains a lot of reflections, the chessboard pattern and a relatively small distance between the views improved the quality of the result. *Buldozer* contains a lot of small details that are clearly separated from the yellow construction of the model, which causes higher variance values when mixing nearby pixels. *Lego* dataset is filled with a single-color area where, for example, the thin edges or details are hard to detect, and the pixels interpolated from the surrounding area might yield lower variance. The distance between *Lytro* cameras is small, so the result was expected to be better. However, due to the technical drawbacks of the camera, the input images contain subtle noise that negatively affects the evaluation. *Pavilion* contains both similar colored areas and complex objects with many details, but the distance between

cameras is large. Fig. 3.18 shows the elapsed time of focus map generation and the final composition of pixels from each dataset.



(a) Reference image is taken from the input light field.

(b) Rendered view shows missing details.

(c) Zoomed reference contains the edges of the bricks.

(d) Zoomed render shows vanishing details.

Figure 3.17: Reference images are placed in the left column. Right column contains rendered reconstructed images with zoomed detail of interpolation artifacts caused by incorrect focus distance estimation in the affected pixels.

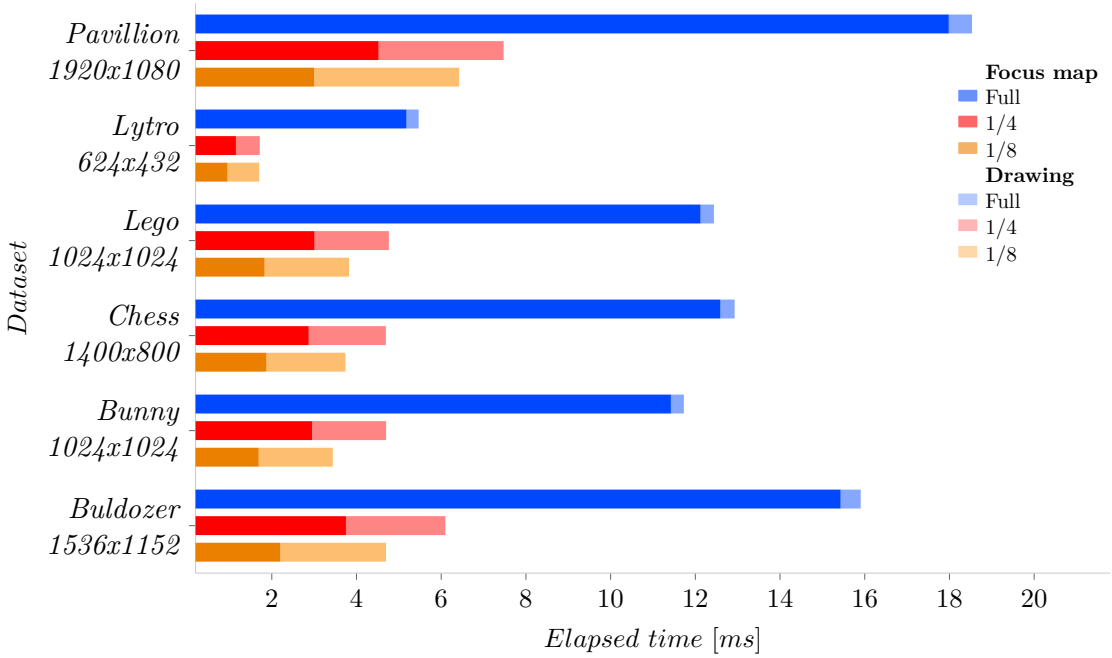


Figure 3.18: Elapsed time of the focus map generation and drawing which depends on the focus map and resulting image resolution respectively. Full, 1/4 and 1/8 sized focus map is used in these measurements. The drawing time slightly increases when using a smaller focus map, most likely due to coordinates interpolation in texturing units due to resolution mismatch.

3.4.3 Focus Map Resolution

One of the key features of the proposed method is the option to separate the focus map generation from the interpolation of the final result. Fig. 3.19 shows how the reduction of the focus map size affects the computation time and visual quality of the final image.

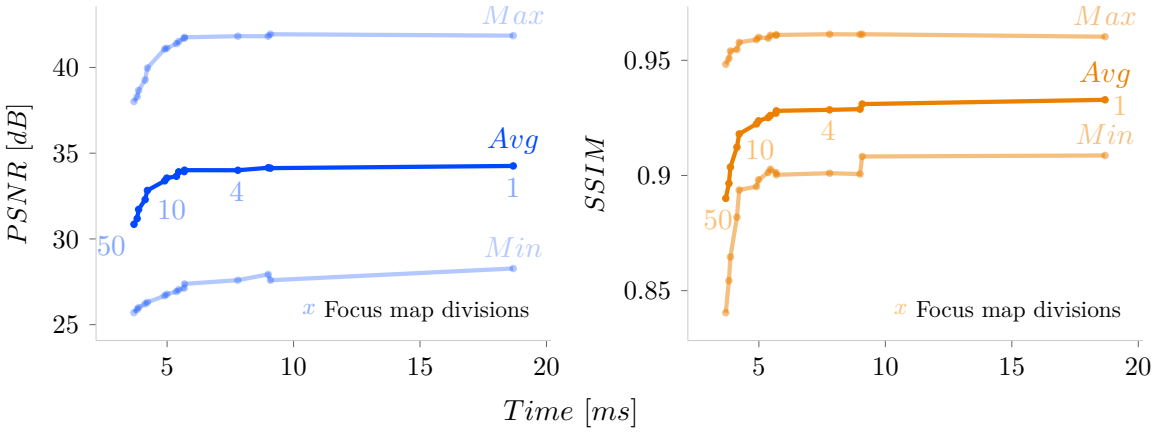


Figure 3.19: The chart shows the relation of visual quality, computation time, and amount of focus map dimensions division. The results are averaged from all tested datasets.

Surprisingly, the quality does not decrease rapidly even with a significant focus map downscaling. In certain cases, the quality even improves because some areas with incorrect focus distances are smoothed due to filtering caused by resizing. Too small map size can lead to the same focus distance on nearby objects that might not lie in the same distance, causing out-of-focus artifacts as shown in Fig. 3.20.

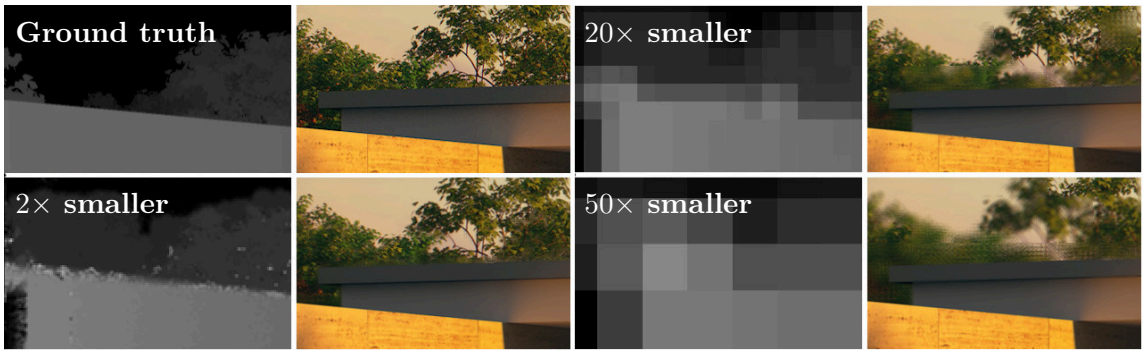


Figure 3.20: Focusing artifacts caused by low resolution focus map. The first image shows reference image with generated depth map. The other ones are focus maps with their dimensions divided by 2, 20, and 50 with the final rendered results.

3.4.4 Focus Range Density

Increasing the number of search samples when iterating over the focus distances in a given range does not significantly help the visual quality and unnecessarily slows down the computation unnecessarily; see Fig. 3.21. 32 samples proved to be an optimal choice for most of the datasets. The most significant difference in quality was measured on *Pavilion* dataset which has the largest depth range. That is the only case where a denser search is necessary. A denser search is needed, especially when the objects in the scene are linearly distributed over the whole depth range.

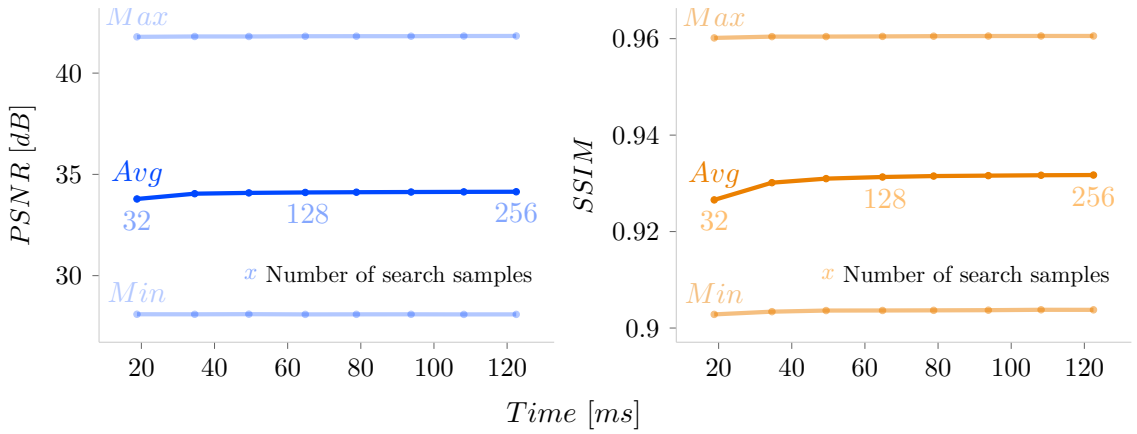


Figure 3.21: Effect of focus range scan density on visual quality is shown.

3.4.5 Camera Grid Sampling Distance

The experimental results in Fig. 3.22 show how many images need to be sampled for the resulting pixel sum. The sampling window in the input grid gives optimal results when having a radius about 2 grid views wide. The wider radius leads to more texture reads and excessive memory access, which slows down the computation most. Surrounding images from the grid in distance from zero to the sampling distance radius are taken into account during the interpolation. When the radius is too wide, images from distant places in the grid might add unwanted ghosting artifacts in the final result. The images capture the scene from a different angle than expected.

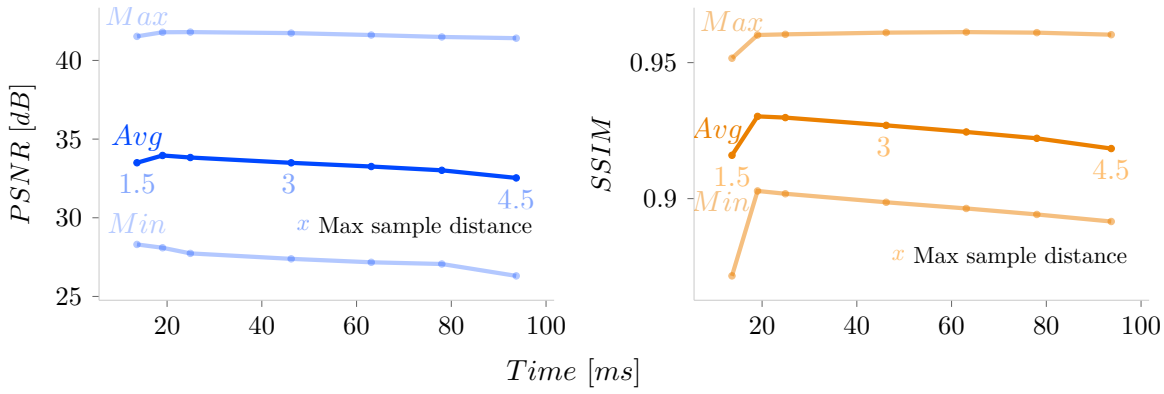


Figure 3.22: Effect of maximal sample distance to quality and computation time is shown.

3.4.6 Camera Grid Parameters

The *Pavilion* dataset was used to measure the relation between the visual quality of the reconstructed view and the distance between cameras with various focal lengths. The distance between cameras, field of view, total depth range in the scene, and position of the camera grid in the scene affect the quality of the reconstruction; see Fig. 3.23.

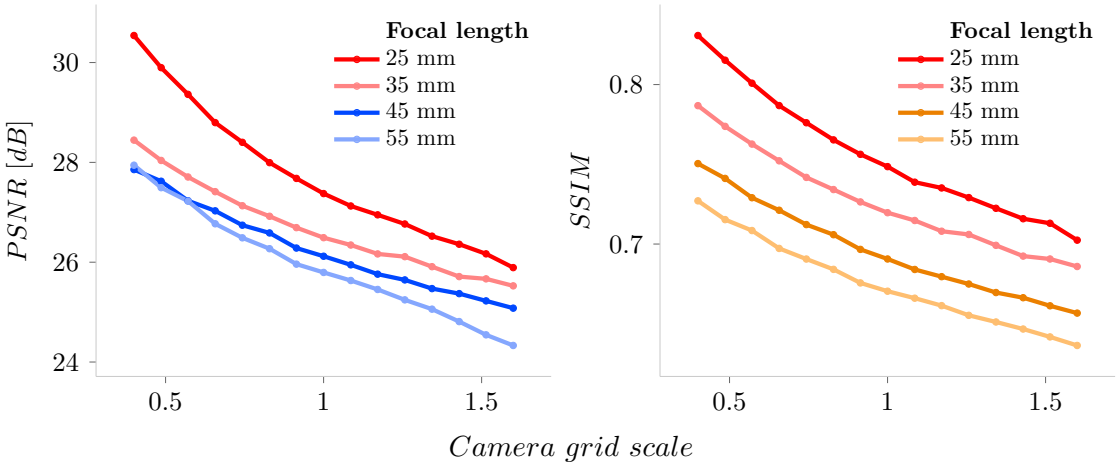


Figure 3.23: The camera grid contains 8×8 cameras and is initially 2 m wide in the scene-space. The first visible surface is about 1 m away from the grid, and the farthest visible spot excluding the sky is about 90 m away. The camera grid is uniformly scaled up and down to change the distance between cameras. The meters units are used according to the world space of the scene.

The camera setup used in the scene can be viewed in Fig. 3.24. With an increasing space between cameras or a decreasing field of view (increasing focal length), the differences between views increase, and the interpolation is more prone to visual artifacts. On the other hand, the farther the cameras are, the more freedom is gained for the virtual camera. This issue can be solved with denser sampling [49], providing more views in the grid, increasing its dimensions. However, this leads to higher memory and bandwidth requirements.

Figure 3.24: The size of the grid (red circle) in *Pavilion* scene and the value of field of view was animated and resulting reconstruction quality was measured. Two views from the corners of the grid using 25 mm focal length are placed in the middle and 55 mm ones at the bottom. The difference between views is bigger in the 55 mm version.

3.4.7 Comparison to Other Methods

An accurate comparison of performance with state-of-the-art methods is complicated due to different methodology and outputs. The proposed method generates a focus map for the new synthetic view used in the rendering stage. The process can be roughly compared to depth or disparity map estimation algorithms. Tab. 3.1 is an indicative overview of the computation times of this stage.

Method	Architecture	Resolution	Time
Proposed	RTX 2070	1920×1080×64	18 ms
[27]	Tesla C2050	640×480×2	16 ms
[18]	E3-1245 V2	541×376×9	1.5 s
[128]	i7 2.8GHz	512×512×49	13 min
[54]	i7-6700k	512×512×49	0.8 s
[59]	Quadro M1000M	1920×1080×45	1.58 s

Table 3.1: The table contains an overview of computation times of state-of-the-art depth or disparity estimation methods from light fields.

The core of the method, consisting of the focus range scan, was first evaluated. The improvements, such as saddle and border detection, block sampling, or map filtering, were then additionally measured to find out how efficient they were. A side-by-side visual quality comparison with state-of-the-art methods is shown in Fig. 3.25.

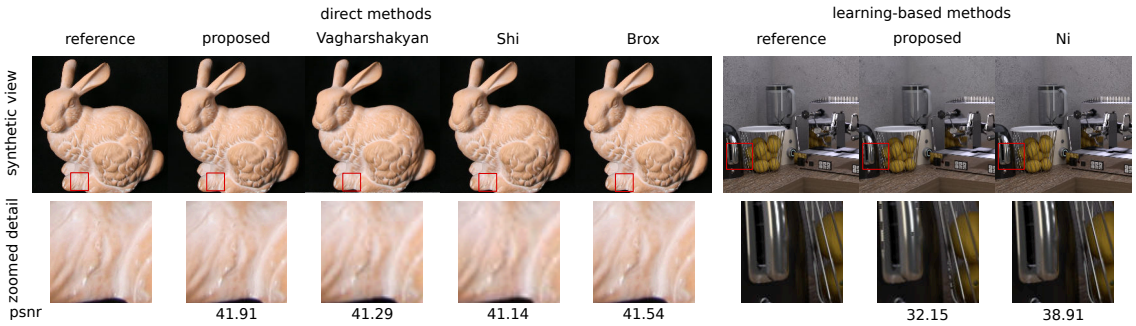


Figure 3.25: Side-by-side visual comparison with other state of the art methods, rendering a new synthetic view. The proposed method without improvements outperforms other general methods but does not reach the same quality as learning-based methods. The results of direct methods do not show any significant differences and are almost identical with difference below 1 dB from the proposed method. The proposed method produces the sharpest result. The proposed method was compared to Vagharshakyan [266], Shi [239], Brox [41], and Ni [196].

Methods capable of producing the synthetic view directly from the images were chosen for the evaluation. The proposed method outperforms other similar approaches. View reconstruction on *Bunny* dataset, using the biggest competitor, the shearlet approach [266], measured on GeForce GTX™ Titan X takes 5 s, which is not suitable for real-time rendering.

The proposed method without further improvements does not reach the same visual quality as the newer learning-based methods [196] when measured on the same dataset used in the original paper, but slightly outperforms older methods [134] (indirect comparison on

Kitchen and *Museum* datasets, difference about 1 dB [196]). However, the proposed method does not depend on the training process, is faster, and is less memory demanding.

Measurements show that the new proposed method is comparable to the other published algorithms in terms of visual quality, reaching a performance suitable for real-time rendering. Fig. 3.26 shows how saddle detection improved the results, compared to the original scan.

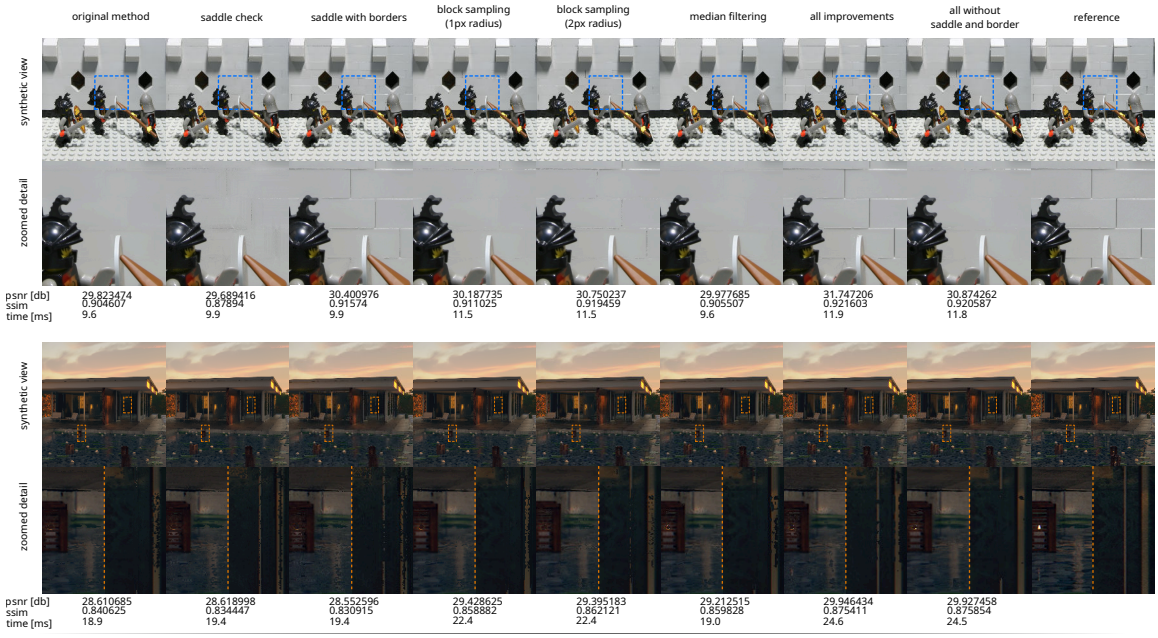


Figure 3.26: The figure contains visual comparison of the proposed improvements, captured on *Lego* and *Pavilion* datasets. Saddle and borders detection leads to huge improvements in visual quality in *Lego* dataset but creates additional noise and artifacts in other ones. Block sampling, using texture interpolation, was used (method *b)* in Fig. 3.8. Two combinations of the improvements were also measured.

The saddle detection and median filtering bring about quality improvement without significant performance loss; see Tab. 3.2. How this improvement made the method comparable even with deep learning approaches is visible in Tab. 3.3. The block sampling patterns are further referenced as *bA, bB, bC* according to Fig. 3.8. Fig. 3.27 shows the best-settings quality improvements for each scene. Fig. 3.28 shows the comparison between the proposed pixel sampling patterns. The automatic focus range detection shows very good results, measured on the data annotated by human; see Fig. 3.29.

Method	Time [ms]		
	1920×1080	**1024×1024**	**624×432**
raw	18.9	9.6	3.2
saddle	19.4	9.9	3.3
borders	19.4	9.9	3.4
bA	30.9	15.7	5.4
bB	22.4	11.5	3.9
bC	43.4	21.1	5.9
median	19	9.6	3.2
bB+m	24.5	11.8	4.1
bB+m+s	24.6	11.9	4.1

Table 3.2: The table contains absolute computational times of focus map generation for all improvements and three common light field resolutions in the tested datasets. The raw label marks the per-pixel scan without the proposed improvements. The time depends only on the resolution of the dataset and does not depend on the content of the scene.

Method	PSNR [db]
film (4×4 images) [217]	42.73
proposed (median+block)	42.62
film (2×2 images) [217]	42.55
Jiang [57]	42.35
Chlubna [9]	41.91
Brox [41]	41.54
Vagharshakyan [266]	41.29
Shi [239]	41.16

Table 3.3: A qualitative comparison with other methods is presented here. PSNR was measured on the *Bunny* scene and compared to other results. The best improvements were used in the proposed method.

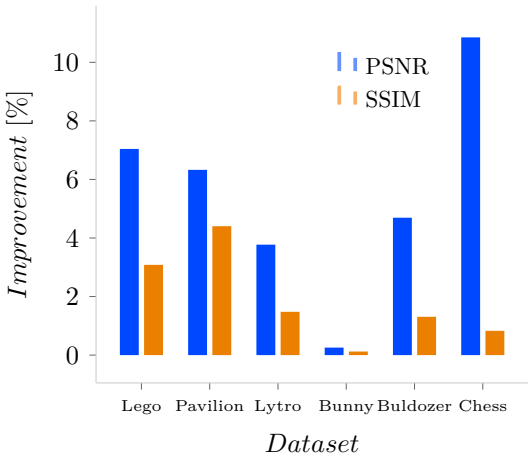


Figure 3.27: The figure shows the best visual quality improvements for each scene in the dataset. The improvement value is relative to the per-pixel scan without additional operations. The complex scenes show higher quality gains than the simple ones, but all scenes are improved.

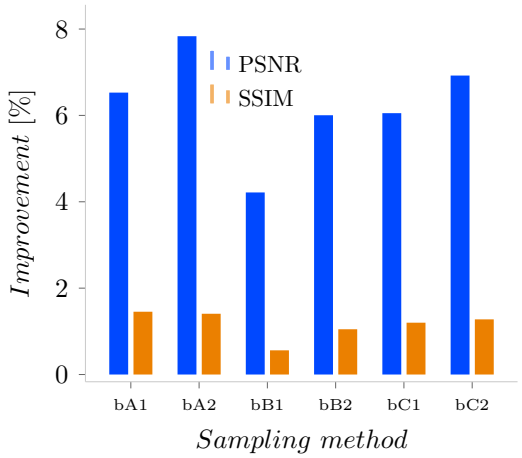


Figure 3.28: The chart shows the visual quality comparison of the three block-sampling patterns. Two distance variations were compared, the first samples the closest pixels, and the second the pixels that are one pixel further away than the first according to the sampling pattern.

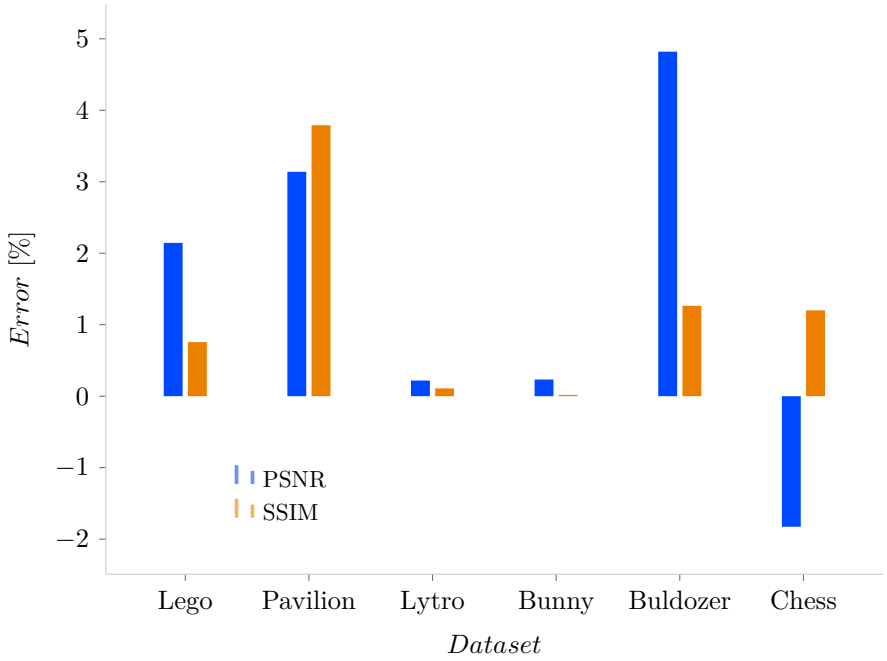


Figure 3.29: Automatic focus range detection is compared to manual focus set according to subjective visual evaluation of the result. The average error of automatic detection using the PSNR metric is 1.46% and 1.19% using SSIM. The detection was able to outperform the manual setting in the case of the *Chess* dataset, measuring PSNR. This was probably caused by the inaccuracy of the PSNR metric according to human visual perception.

3.5 Focusing Performance Optimizations

The generation of the focus map can be accelerated to produce a better quality/speed ratio [10]. The optimal computational efficiency mentioned in Hypothesis is the main goal of this research. This section describes the proposed optimization of the above-described light field rendering method; see Fig. 3.30.

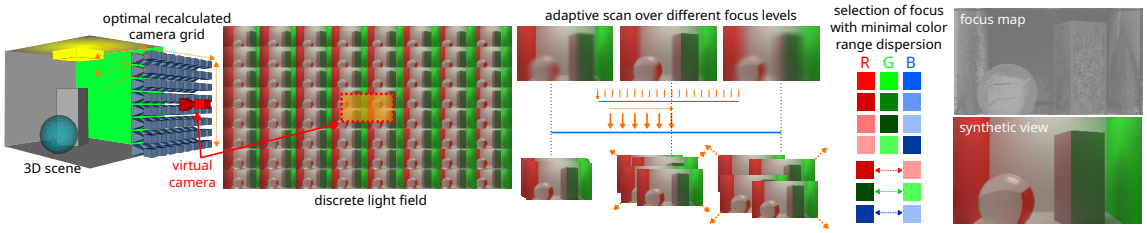


Figure 3.30: Real-life or synthetic scene is captured by multiple cameras which are optimally positioned according to the proposal. The captured images, closest to the virtual camera, are mutually shifted to simulate light field focusing. Multiple focus distances are scanned using a novel optimal strategy with an improved color dispersion metric. Optimal focusing for each pixel is stored in a focus map which is filtered and used to synthesize a novel view. The result is produced by the proposed method with *Cornell box* scene.

3.5.1 Focus Range Scanning

Scanning of the focus range and looking for the minimal color dispersion was previously [8, 9] carried out using a brute-force approach. A fixed number of steps with constant spaces was used. This brute-force scan has two disadvantages. First, it can skip the optimal value due to the constant spacing of the samples. Second, it is computationally demanding because each sample involves GPU memory access operations that slow down the process. The goal of this research is to find a better strategy that adaptively scans the relevant parts of the range and skips the parts that do not contain any potentially desired values. Several alternative scanning methods are proposed and tested:

- **BF** (brute force) is a simple scan with fixed number of steps and constant spaces.
- **BFET** (brute force with early termination) is the BF scan that stops if the last optimal value does not change in a defined number of steps.
- **VS** (variable step) shrinks the step size when a new optimal value is found and grows otherwise.
- **VSET** (variable step with early termination) is the VS scan that stops if the last optimal value does not change in a defined number of steps.
- **RAND** (random) scans the range at random positions with an early termination mechanism.
- **TD** (top down) starts with three samples, and in each iteration scans the center positions of the remaining spaces between the previously sampled positions.

- **HIER** (hierarchical) uses bisection method to scan the range sparsely and shrink the scanning interval in each iteration.
- **DESC** (gradient descent [100]) starts at multiple uniformly selected positions in the range. Each scan branch uses a variable scan step that changes according to the difference between the last focus value and the new optimum. The best value of all branches is selected in the end.
- **PYR** (pyramid) performs the first scan on downsampled and blurred data. A tight area around the found optimum is scanned again on the full-resolution data. The downsampling might help to identify the area that most likely contains the desired values, and large disparities or single colored areas might be easier to detect [263]. This method can exploit texture mipmapping.

Fig. 3.31 describes the methods. Stochastic variations of the proposed methods were tested too, but they showed too much thread divergence on GPU and were too slow.

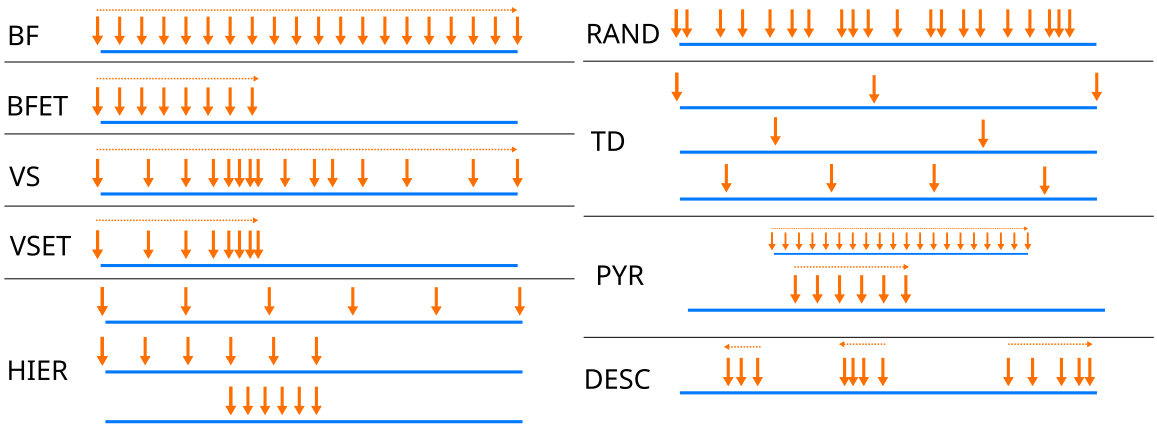


Figure 3.31: Scanning methods compared in this study are depicted in the figure. The blue line marks the whole focus range. The orange arrows are samples in the scan. Each method scans the range with different number of steps and at different positions.

3.5.2 Color Dispersion Metric

A statistical dispersion metric is used when the colors of the pixels sampled are compared. This metric is used to estimate the probability that the sampled pixels belong to the same spot in the scene. The assumption is that the colors of such pixels would be similar. The color function C in Eq. 3.5 for a specific pixel and synthetic view position is a mapping between the focus level f and the light field image grid coordinates $\mathbf{st}$ into the m-dimensional color space (typically $m = 3$ for RGB or YUV spaces):

$$\mathrm{C}(\mathbf{st}, f) : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^m. \tag{3.5}$$

An array of colors for all images $\mathbf{X}(f)$ from $S \times T$ sized grid is processed in Eq. 3.6 as:

$$\mathbf{X}(f) = [\mathrm{C}(\mathbf{u}, f)] \qquad \mathbf{st} \in [1 \ldots S] \times [1 \ldots T]. \tag{3.6}$$

The optimal focus distance f_o in Eq. 3.7 is found from the focus range $[f_s, f_e]$, based on the minimal color dispersion value $\mathrm{disp}(\mathbf{X}(f))$:

$$f_o = \arg\min_{f \in [f_s, f_e]} \{\mathrm{disp}(\mathbf{X}(f))\}. \tag{3.7}$$

This study contains a comparison of several dispersion metrics that differ in quality and computational complexity:

- **VAR** (variance) is standard statistical variance.
- **RANGE** (total range) measures the maximal and minimal distance from the origin, and their difference is marked the dispersion.
- **ERANGE** (elementwise range) is a metric where maximal and minimal values are stored for each color vector element. The distance between the maximal and minimal vectors marks the dispersion.
- **MAD** (mean absolute difference): would be too complex to compute, so a simple approximation is used. The difference of the new color and the last one and the difference of the new color and one sample from the history are added to the final dispersion.

Interquartile range metric [63] was also considered but not measured because the preliminary test showed that this metric is computationally too demanding and the results in terms of visual quality were not promising.

3.5.3 Pixel Sampling

Memory requirements of light fields are one of the main issues in the rendering pipelines. Encoding in video formats was previously proposed [8] to stream the necessary packets in the GPU memory, where the frames are decoded in the HW decoders. Most decoders, such as NVDEC, return frames in YUV format. Conversion between color spaces might bring additional overhead. The attributes of YUV might be exploited to achieve better results than in standard RGB. Given the assumption that most of the materials in the real world are partially diffuse and specular, the color of the pixel at the same spot on the scene geometry should not change with the changing camera position. The luminance of the pixel can change. The Chebyshev distance [227] proved to be the best metric to measure color similarity in focus map generation [9]. This distance function can be adjusted to attenuate the luminance component in Eq 3.8 as:

$$d = \max\left(\left|c_Y^A - c_Y^B\right| \cdot w_Y, \left|c_U^A - c_U^B\right|, \left|c_V^A - c_V^B\right|\right), \tag{3.8}$$

where d is the final distance, c^A and c^B are the input colors with subscripts $c_{Y,U,V}$ representing the color components and w_Y is the attenuation weight for the Y component distance.

Additional strategies for sampling the texture outside its borders were proposed, as the shifted textures always end up being sampled outside. Blending of the pixels at the border, where the radius grows with the distance from the border and alternating of the last pixels instead of copying of the last one might create more distinctive details.

3.6 Experimental Evaluation with Novel 4K Dataset

The implementation uses CUDA kernels for the focus map computation. Visual quality was measured using PSNR and SSIM metrics. Quality was measured relative to the basic BF method and averaged. The number of scan steps was set to 32 according to previous measurements [9], as it ensures the optimal speed/quality ratio. Decreasing the steps leads to quality loss with only linear speed-up. Additional no-reference metrics LIQE [296] for standard artifact detection and NIQSV+ [259] for multi-view image synthesis evaluation were used for further analysis of the results. MNSS [92] metric designed for DIBR (depth-image-based rendering) was also tested, but the results did not correspond to the actual quality of the novel view. The DIBR artifacts are apparently of a different nature than artifacts appearing in light field rendering. Measurements were carried out on all scenes of the dataset, unless explicitly stated otherwise. Five reference views for each measurement were linearly selected, with constant spaces, from the reference dataset grid and evaluated. The dataset was generated by the method described in Section 3.7.

3.6.1 Important Parameters Identification

Initially over six million test combinations were considered, but due to the size being unreasonable, only the most promising parameters were selected for full testing. A quick test was conducted on *street animation* scene to eliminate parameters that did not show significant improvements. The scene was selected because it resembles an ordinary real-life photo and contains various materials.

Reduction of Scan Methods

RAND, VSET, VS and BFET methods are considered to be only minor optimizations. Only VSET was selected for the further measurement, according to Fig. 3.32.

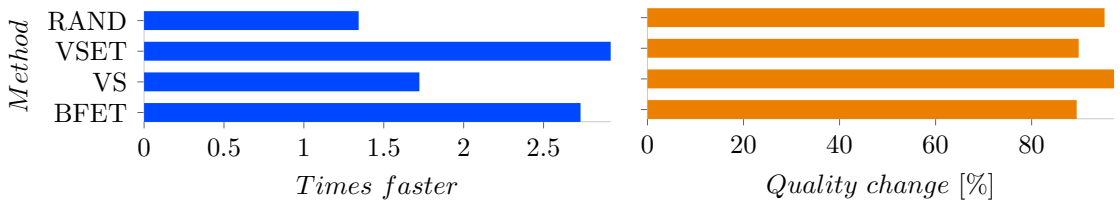


Figure 3.32: These four similar scanning methods were compared to select only one for further measurements. The quality does not change significantly, but the speedup is the best at VSET. The measurements are relative to the base BF method.

Although the quality metrics were not in favor of RAND, this scan method might be useful for early preview purposes similarly to progressive path tracing, where the amount of noise decreases with every new sample [287]. The advantage of RAND is that the error distribution is uniform throughout the image [178]. This method can transform the typical light field ghosting artifacts to classic, per-pixel, color noise. Other methods usually leave whole areas distorted, so the area-wise error is more visible to human observer than the pixel-wise error; see Fig. 3.33. Perceptual LIQE metric score for RAND was 1.97 compared to 1.73 for VSET and 3.382812 for the reference image. NIQSV+ score was 7.0 for RAND and 6.9 for VSET.

(a) Reference image is taken from the input light field.

(b) RAND focuses the scene but adds noise everywhere.

(c) VSET leaves some parts of the scene out of focus.

Figure 3.33: The figure shows how VSET scan can reconstruct many small details but can contain large areas with artifacts. RAND contains more pixels with error, but the error might not be visible because it is distributed uniformly. The images are zoomed parts of the *street animation* scene.

Map Filtering

The final focus map usually contains noise. The map was filtered by three methods: median filter, symmetric nearest neighbor [101], and Kuwahara filter [150]. Median and Kuwahara produced better quality results. Kuwahara filter was identified as the best option because the same quality of the result could be achieved in half the time of the previously proposed median filter.

Color Sampling

The pixels are sampled by blocks to obtain information about the surrounding details. Previous measurements showed that pixels can be optimally sampled in certain patterns with a kernel with a width of 2 pixels [9]. Only 5 pixels are sampled in the distance defined by the radius. One pixel in the middle and four interpolated, each between the four pixel coordinates in the corners. The measurements on 4K data showed that the visual quality improved with increasing block radius up to 10 pixels. The quality reached an improvement of 13% compared to the single-pixel sampling. Experiments on various sizes of the data showed that the radius can be linearly adjusted according to the total number of pixels, for example, 2.5 for FullHD resolution.

The change of color spaces does not significantly affect the result. It can be safely assumed that the proposed method works with the same efficiency in both RGB and YUV. YUV formats, such as widely used NV12, store the Y values in a separate array from UV. Reading the whole YUV triplet would cause inconsistent read operations on GPU which might negatively affect the performance. The measurements showed that the color distance can be replaced by the simple difference of Y components of the colors with the loss of quality by 3%. The proposed weighted YUV Chebyshev distance, see Eq. 3.8, slightly outperforms the other mentioned approaches with the attenuation weight set to 0.25 which was empirically discovered. The quality improvement is 0.2% compared to standard RGB and 0.8% to YUV Chebyshew.

Other Parameters

Measurement of the behavior of the out-of-texture sampling showed that the computational time does not change significantly. The visual quality of standard wrapping, mirroring, and solid border modes is the lowest. Clamping, blending, and alternating produced similar results, with alternating slightly outperforming the rest by 0.34%.

The effect of the order of the color distance ($distance^{order}$) function was measured, but the effect on the result is not statistically significant. The focus range was originally scanned linearly. The order of non-linear transformation ($range^{order}$) was also measured since the details that are far from the camera might not be as dense and might need a sparser scan than the objects nearby. This assumption was not confirmed, and the linear range showed the best results. Both orders were kept at the value of 1 in all other experiments.

The average quality measured in all tests showed that the input views can be reduced to only 4 nearest neighbors to the synthetic view, compared to the previously proposed 16 views. The quality loss is only 0.3%, but the reduced number of texture read operations drastically speeds up the process 10.6× when measured over all the approaches mentioned.

3.6.2 Color Dispersion

The dispersion metrics were compared by averaging all measurements with all scanning methods; see Fig. 3.34. The previously proposed VAR metric for the evaluation of color dispersion can be replaced by ERANGE which is 4.14× faster and reaches better quality than VAR by 3%.

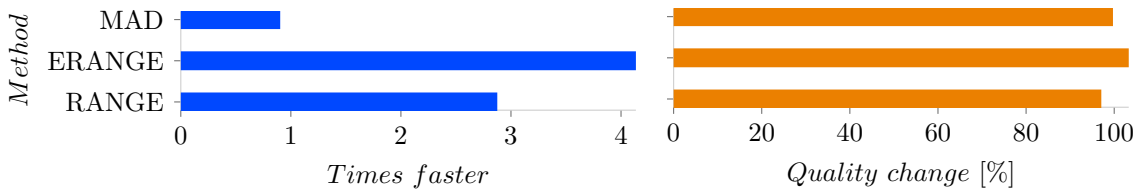


Figure 3.34: The color dispersion metrics were compared over all measurements. The ERANGE metric outperforms all other approaches in time and quality. MAD metric brings no improvement. The measurements are relative to the base VAR method.

3.6.3 Focus Scanning

The comparison of the scanning methods is performed with the ERANGE metric, but the results are similar even with the other ones; see Fig. 3.35. The PYR method reaches almost the same quality as the base BF method, the quality loss is 2.86%, but it is 1.74× faster. The TD method is the fastest, 2.96× faster, but the quality loss is 12.83%. HIER reaches 1.06× speedup with 3% quality loss.

No-reference metrics were also used to evaluate how the artifacts manifest in the best scan. NIQSV+ and LIQE scores for BF and PYR were 5.44, 5.41 and 1.86, 1.84 respectively. This shows that the perceptual distortion of the PYR scan is minimal compared to the BF.

Fig. 3.36 shows the common errors in the results. Although the number of scanning levels can be reduced to speed up the original BF method, the constant scan leads to a crude focusing approximation where certain ranges are never scanned at all. The proposed scan

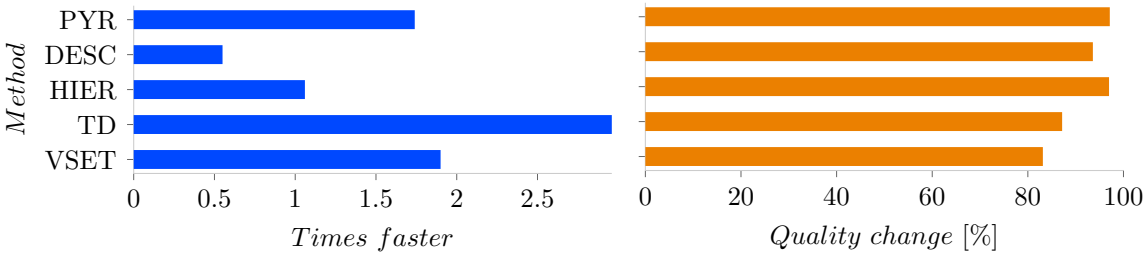


Figure 3.35: The scanning methods were compared, showing that all methods are significantly faster than the base BF. The PYR method outperforms all other approaches in time and quality. The measurements are relative to the base VAR method.

methods are less prone to vanishing details and ghosting due to the possibility of denser scan when necessary.

(a) Reference image is taken from the input light field.

(b) BF method creates incomplete surfaces.

(c) PYR significantly mitigates the artifacts.

Figure 3.36: Parts of the most challenging *high frequency* scene are shown on the pictures. The number of steps of the BF method was lowered to reach the same time as PYR.

3.6.4 Optimal Setup

According to the above measurements, the optimal parameters are identified and the best method for focus map generation is proposed with parameters listed in Tab. 3.4. The comparison with the previously proposed and other state-of-the-art methods is shown in Tab. 3.5.

Scan	Metric	Views	Filter	Block	Color	Texture
PYR	ERANGE	4	Kuwahara	10×10	YUVw	ALTER

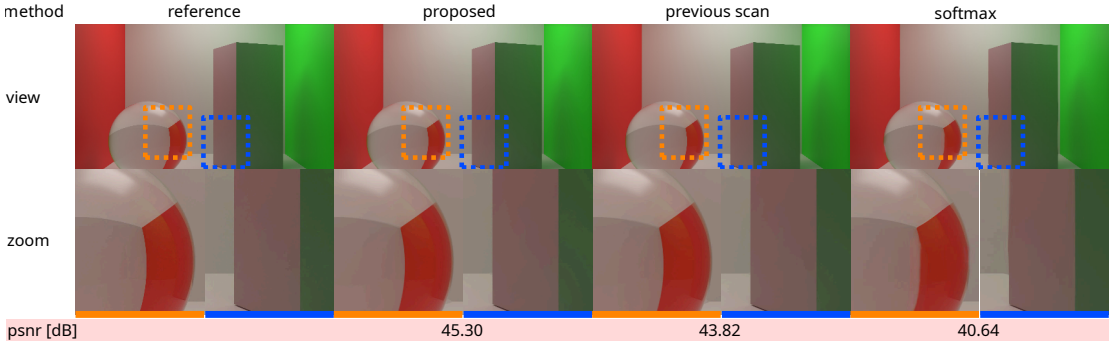
Table 3.4: The best-quality and fastest computation parameters were identified.

Method	Views	PSNR [dB]	SSIM	LIQE	NIQSV+	Time [ms]
proposed	4	30.26	0.86	5.41	1.84	51.21
previous scan [8]	4	30.09	0.872	5.32	1.79	396.85
previous scan [8]	16	30.1	0.86	5.44	1.82	1769.12

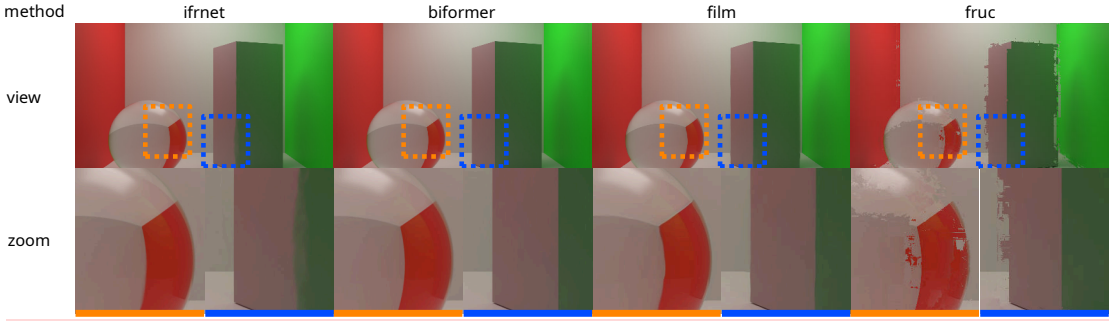
Table 3.5: Previous focus map generation method is compared to the improved proposal.

Fig. 3.37 shows a visual and measured comparison of the proposed method and other recently published approaches. Furthermore, the FLAVR [135], ST-MFNet [60] and

SAVFI [57] methods were tested but required more views than 4 closest to produce a reasonable result. Instant NGP [190] was able to use 4 views, but the result was still blurry even after minutes of processing.



psnr [dB]	45.30	43.82	40.64
ssim	0.988	0.989	0.985
NIQSV+	8.13	7.77	6.9
LIQE	2.75	2.12	1.62
time	57.78 ms	361.89 ms	4.69 s
memory	1.2 GB	1.18 GB	5.36 GB
resolution	3840x2160	3840x2160	1920x1080



psnr [dB]	42.43	42.26	39.1	28.9
ssim	0.99	0.989	0.982	0.966
NIQSV+	6.7	7.98	6.33	5.42
LIQE	1.43	1.6	1.42	1.52
time	1.2 s	33.15 s	65.51 s	110 ms
memory	4.75 GB	6.54 GB	7.37 GB	1.08 GB
resolution	1920x1080	1920x1080	1280x720	3840x2160

Figure 3.37: View interpolation methods were compared on *Cornell box* scene. The other methods are: previous scan [8] for focus map generation, deep interpolations such as softmax [197], ifrnet [144], biformer [204], film [217], and optical-flow based fruc by NVIDIA. All methods were measured on GPU. Inputs of some methods had to be downscaled.

The 4 closest images are problematic for the 3D and neural [274] reconstructions, as the 3D information is difficult to obtain from only four views positioned on a plane. For example, Colmap library, Meshroom, or 3DF Zephir were unable to correctly position the cameras for dense point-cloud reconstruction. All these results show that the proposed method is valuable for real-time processing as it saves memory, produces a high-quality synthetic view, and is significantly faster than other methods. The previous focus map scan used 16 nearest views. The implementation of the previous method used in the experiments was adjusted to use 4 views, otherwise the speed-up would be even higher.

The memory and time requirements of the proposed method are lower than state of the art. To prove that the method reaches the same level of visual quality as existing methods, a qualitative comparison was conducted on different existing datasets. A zoomed detail of the *chest* scene is compared in Fig. 3.38 where many methods ignore the details on the

crystal and blur them. Optical-flow-based fruc, on the other hand, unnecessarily highlights the details.

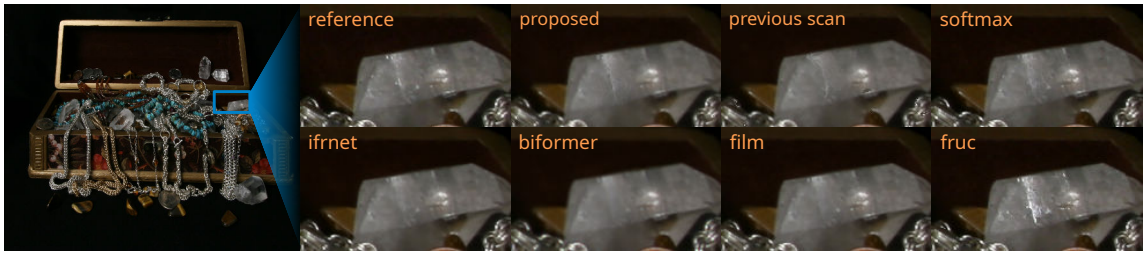


Figure 3.38: A problematic zoomed region of *chess* [267] scene is compared across the interpolation methods. The reflective material of the crystal with the small crack-like shiny details is hard to reconstruct.

Fig. 3.39 shows full novel views views from different scenes. The results look visually the same with minor differences.

The proposed method can produce artifacts similar as the existing methods but is stable. Deep learning methods can fail when the pretrained model does not contain the necessary information for the specific kind of scene. The example of such an issue is especially visible in the *simple setting* scene with softmax and ifrnet. Although deep learning methods can sometimes produce higher-quality results, the proposed method is more stable and lightweight.

3.6.5 Scene Types

The original dataset proved to be a good benchmark of various types of scenes. The best quality, PSNR above 40 dB and SSIM over 0.95, was measured at *Cornell box*, *low frequency* and *simple setting* scenes. The worst quality, PSNR below 25 dB and SSIM below 0.75, was measured at *high frequency* and *large depth* scenes. It can be assumed that two crucial parameters regarding the resulting quality of the proposed method exist. The first is the frequency of the features in the scene, as the sampling can miss the high frequency patterns according to the Nyquist–Shannon theorem [49]. The second is the depth range, where the wide range is more difficult to properly scan without skipping important details in the scene. Both parameters need to be high to decrease the quality. For example, *low frequency* contains a large depth range but belongs to the best results. The method is efficient with reflective materials, volumetric effects, and single-colored areas, which are often considered problematic for image-based rendering methods. Surprisingly, the *reflective* scene reached slightly better quality of the novel view than its counterpart *diffuse*. Both scenes contain the same geometry and differ only in material.

The views need to be synchronized in time [272] and captured by the same type of camera. The proposed method can produce a novel view even from temporarily unsynchronized views with different brightness levels. The color similarity metric ensures that the least error value is produced, so the unsynchronized results are simply motion blurred.

Several filters including CLAHE contrast [242], edge detection, sharpening, histogram equalization, bilateral filtering, sinusoidal color transformation, etc. were tested for both denoising and detail highlighting. The quality remained the same, and the filtering did not aid the algorithm. However, this proved that the method is robust enough and can tolerate such filters and applied effects on the views.

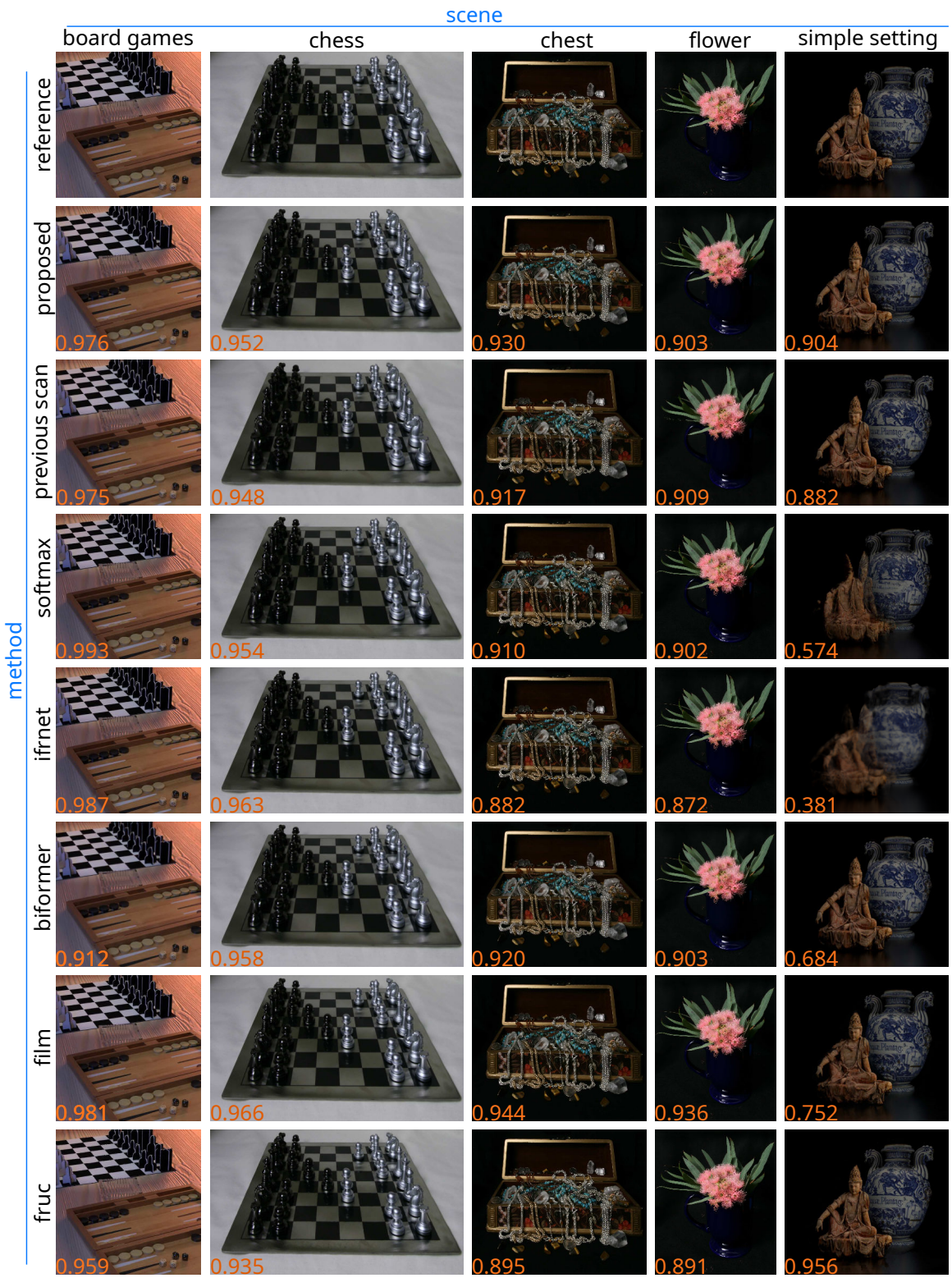


Figure 3.39: Existing view interpolation methods are qualitatively compared using scenes from different datasets: *board games* [113], *chess* [267], *chest* [267], *flower* [267], and *simple setting* (original dataset). The orange numbers in the bottom-left corners are SSIM results.

3.7 Novel Dataset Capturing Method

This section proposes an optimal method for capturing light field data [10]. It supports the Hypothesis by creating optimal dataset for the proposed method. This data set ensures the best visual quality of the results produced by the proposed rendering.

Narrowly spaced data, for example, taken by plenoptic cameras, are useful for refocusing of the final photo at different distances [286]. They can be represented as a grid of images with small spaces or by a focal stack, where the images are taken from the same position but differently focused [152]. Widely spaced data allow for 3D movement in the scene [283]. Widely spaced discrete light field data are the main topic of this research.

The shape of the camera grid used is usually defined by the technical limitations of the rig or by subjective visual evaluation of the results. Existing datasets usually use constant spacing between cameras [158], or different fixed spaces in both axes depending on the capturing mechanism [109]. Reconfigurable camera arrays also exist, where camera positions are adjusted based on the rendering quality of the result [295]. The cameras are usually positioned to achieve a defined overlap, for example, complete view overlap at 10 feet from the array [220], instead of depth-aware adjustment. This section proposes a capturing method designed to produce the best results with the proposed rendering method by having the same amount of view overlap in both axes.

Two main parameters can be defined by the user to affect the positioning of the cameras in the grid: the distance from the cameras plane to the geometry of the scene and the amount of overlap between the cameras. These parameters are more intuitive for the user than, for example, size of the camera grid, because they reflect how much of the scene content the resulting views share between each other.

The geometry of the scene is usually not planar. The distance to the geometry needs to be defined as either a distance to the object of interest or to the average position of the whole visible geometry. Occluded vertices or varying density of the geometry in a synthetic scene disallow the method to analyze the geometry itself. A screen-space approach is proposed to solve this issue. In case of a synthetic scene, the scene is quickly rendered from the reference camera with all object materials overridden by a special material, which converts world-space coordinates of the fragments into colors. The average of such a rendered image is a decent approximation of the average position of the visible geometry. One, even a low-quality, depth sensor would suffice for real-life camera grids. The camera grid is then created, taking a reference camera position and orientation as the center of the grid.

Having square-shaped cameras with the same width and height and the same horizontal and vertical spaces between the cameras always results in the same-sized overlapping areas in the views. The aspect ratio between the width and height of cameras in modern cinematography or photography is usually used with a wider width, for example, $16 : 9$ [48]. Regular spaces between such cameras would lead to unequal overlap. Spacing simply scaled by the aspect ratio would ensure that the cameras exactly touch their borders when the overlap is zero but would be unequal when it is greater. The spacing needs to be recalculated based on the input parameters. Standard 50 mm lens was used in the implementation, as it creates the most natural-looking image for a human observer [26]. Fig. 3.40 demonstrates the described issue.

Fig. 3.41 depicts the positions of the cameras in relation to the scene and along with the 2D-aligned view in Fig. 3.42 contains definitions of the variables used below.

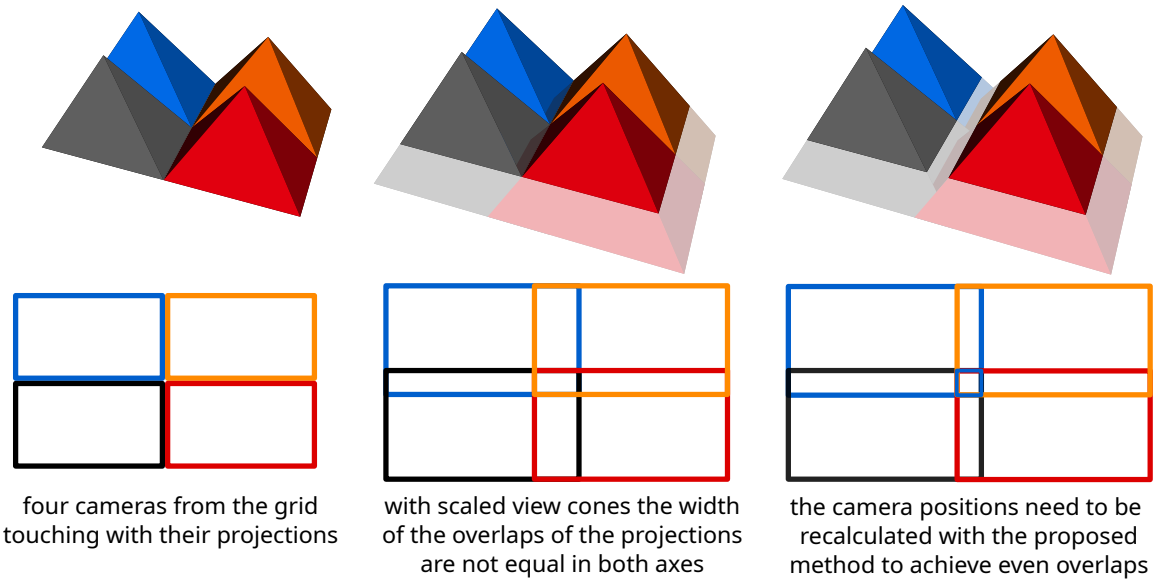


Figure 3.40: The figure demonstrates the problem of overlaping view cones in light field camera grids. A non-uniform aspect ratio, e.g., standard 16 : 9, creates different overlapping areas of the image planes between the neighboring cameras in the vertical and horizontal directions. Recalculation of the camera spacing by the proposed method is necessary to ensure the same amount of overlap in both axes.

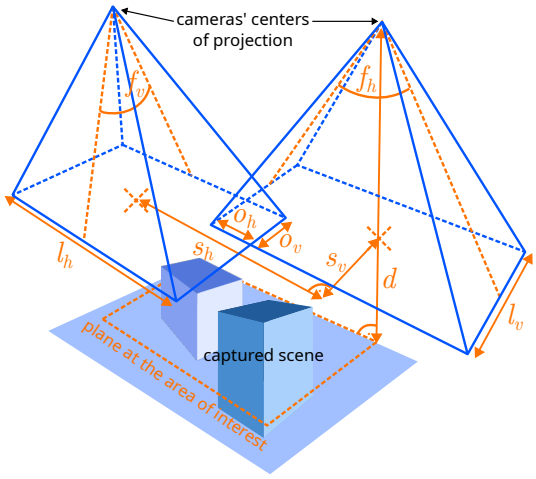


Figure 3.41: Two capturing cameras placed at two neighboring rows and columns are depicted. The s_h, s_v, and d lines are going through the centers of projection. The camera cones overlap in both axes with amount o_h and o_v. The goal of the method is to achieve the state where $o_h = o_v$.

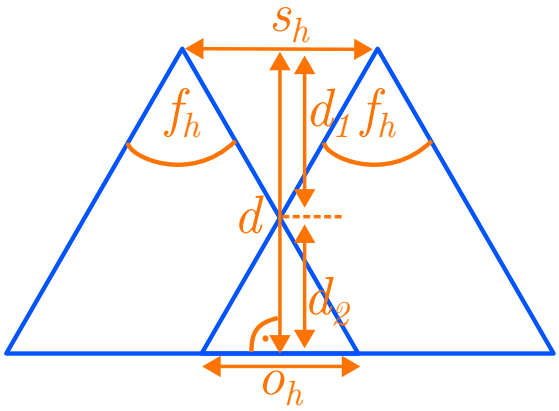


Figure 3.42: 2D view of two neighboring cameras is shown. The spacing s_h between the cameras is the desired result. The ends of the view cones represent the image planes.

Given a horizontal field of view f_h, horizontal l_h, and vertical l_v camera length or resolution, the tangent of half of the vertical field of view f_v can be obtained from

$$\frac{\tan\left(\frac{f_h}{2}\right)}{\tan\left(\frac{f_v}{2}\right)} = \frac{l_h}{l_v}. \tag{3.9}$$

The horizontal distance between cameras s_h can be expressed from

$$\tan\left(\frac{f_h}{2}\right) = \frac{s_h}{2d_1} \tag{3.10}$$

as

$$s_h = 2\tan\left(\frac{f_h}{2}\right) d_1, \tag{3.11}$$

using the distance d_1 from the cameras plane to the closest point of camera frustum overlap. This distance can be obtained from

$$d_1 = d - d_2. \tag{3.12}$$

The distance d is the total distance from the cameras plane to the scene geometry and d_2 is the distance from the aforementioned point of overlap to the scene geometry. Both d_1 and d_2 are unknown but can be obtained by expressing d_2 from

$$\tan\left(\frac{f_h}{2}\right) = \frac{o_h}{2d_2}, \tag{3.13}$$

where o_h denotes the amount of overlap between the cameras. Combining the equations 3.12 and 3.13 expresses d_1 as

$$d_1 = d - \frac{o_h}{2\tan\left(\frac{f_h}{2}\right)}. \tag{3.14}$$

Combining equations 3.11 and 3.14 results in

$$s_h = 2\tan\left(\frac{f_h}{2}\right)\left(d - \frac{o_h}{2\tan\left(\frac{f_h}{2}\right)}\right), \tag{3.15}$$

which can be simplified to the final formula:

$$s_h = 2\tan\left(\frac{f_h}{2}\right) d - o_h. \tag{3.16}$$

The same approach can be used to analogously express the vertical distance between the cameras s_v:

$$s_v = 2\tan\left(\frac{f_v}{2}\right) d - o_v. \tag{3.17}$$

The horizontal o_h and vertical o_v amount of overlap should be the same according to the desired constraint:

$$o_h = o_v. \tag{3.18}$$

This method, described by Eq. 3.9–3.18, was used to produce a novel 4K synthetic dataset containing 22 static and 4 animated light field scenes with the size of the grid 15×15 views. The scenes were selected to cover most of the commonly measured scenarios such as diffuse/reflective materials, wide/narrow depth range, high/low frequency features, single/multiple objects, near/far objects, all-focused/depth-of-field, solid/volumetric objects, and their combinations. The currently existing datasets are usually available in lower quality and do not cover all the mentioned scenarios suitable for benchmarking.

3.7.1 Evaluation

The amount of overlap in the input views can affect the quality of the novel synthetic view. The distance between the cameras depends on the defined minimal overlap value. When the view is not in 1:1 ratio, the minimal overlap is defined on the vertical axis. With constant spaces between the cameras, the views would overlap more on the horizontal axis. This might be redundant. Using the proposed camera distance calculation method ensures that the minimal overlap will be used in both axes.

An experiment was conducted to measure how the quality changes when the aforementioned approaches are used. The 4K *street animation* scene from the dataset was used with the default camera settings documented in the dataset files. Tab. 3.6 shows that the quality does not change significantly. However, the proposed method allows for larger spaces between the cameras, which means that a larger portion of the scene is captured without quality degradation. For comparison, an even larger spacing was measured in the table to show how the quality decreases when the distance is too large. The proposed method ensures the best quality of the result with the largest possible spacing. Note that the spacing difference in favor of the proposed method might be even larger with other camera configurations.

Method	**PSNR [dB]**	**SSIM**	**LIQE**	**NIQSV+**	**width**
proposed	37.00	0.95	3.039	7.09	1.816
const	38.74	0.96	3.139	7.18	1
aspect	37.22	0.95	3.035	7.03	1.778
wide	31.41	0.0.92	2.57	6.87	5

Table 3.6: Various light field camera grids are compared. The proposed method ensures the largest grid width without visible quality degradation. The grid height remains the same in all measurements. When the width of the grid is too large, the quality loss is visible.

3.8 Hypothesis Proof Summary

The presented results complete the proof and solve the set goals. The proof of the proposed Hypothesis not only supports the scientific achievements of the thesis, but also, perhaps more importantly, shows that the light field rendering can be practically used. The proposed rendering method highlights the advantages and mitigates the disadvantages of light fields as much as possible to achieve better quality than existing methods and significant computational speedup.

Based on the results achieved and published, the proposed Hypothesis can be claimed as proved. Each part of the hypothesis was proved as follows:

- *A novel method of light field rendering can be designed and implemented...*
  A reference implementation of the proposed method was created and used in the experiments. The novelty of the method was proved by peer-reviewed publications and acceptance by the scientific community.

- *...that produces all-focused renders directly from input views without exploiting additional depth information,...*
  The method is capable of automatically detecting the best focus distance for each pixel of the result, as described in Section 3.2.1. The result is all-focused, with minimum visible out-of-focus artifacts. The proposal works only with the input images, so no structural information about the scene is necessary, as described in Section 3.2. The necessary focus range of the scene can be detected automatically, as shown in Section 3.3.

- *...having better visual quality, being computationally more efficient, and having less memory requirements compared to the state of the art.*
  The experimental evaluation of the proposal in Sections 3.4 and 3.6 shows that the proposed method achieves better visual quality, higher computational efficiency, and less demanding memory requirements than state-of-the-art methods.

The experimental measurements were conducted on standard datasets and also on a novel original dataset. It is possible that certain data might not be suitable for the proposed method. However, the tested datasets cover most of the scenes commonly used in both industrial and real-life applications. Therefore, it can be assumed that the proposal can generally achieve optimal results.

On top of the hypothesis proof, the following light field rendering advantages, mentioned in Chapter 2, are highlighted in the proposed method:

- **Constant render time**
  The proposed method is based on scanning of the focus range that consists of a fixed number of steps; see Section 3.2.1. The most time-consuming part of each step is the texture read operation, which can also be considered constant-time. This ensures a constant render time, which depends only on the light field resolution or focusing quality settings.

- **Constant memory requirements**
  The method uses only the nearest views of the light field grid; see Section 3.6.1. All necessary memory allocations can be made in advance and no reallocations are necessary.

- **Rendering algorithm independent of scene content**
  The previous two points show that the algorithm performs the same operations on all types of data. The camera position change only affects the selection of the necessary views of the light field grid. The measurements carried out showed that the method is robust and works well for existing standard and novel original datasets; see Sections 3.4 and 3.6. It can be assumed that the method would work for most potential scene types independently of their content.

- **Simple representation as textures**
  The method adopted the widely used format of discrete light field approximation as a set of images. The standard datasets can be used directly without any preprocessing; see Section 3.4.

- **Straightforward acquisition**
  No changes are necessary in the acquisition process. A simple capture of the scene in a predefined grid suffices. The proposed optimal acquisition method follows the same principles and only affects the positions of the cameras; see Section 3.7. However, the method works even for data that were captured without the proposed guidelines, although not necessarily achieving the highest possible quality.

- **Easily implementable on parallel architectures**
  Light field rendering is expected to be used primarily on GPU to make it available in graphic applications. The proposed method is designed to fully utilize GPU parallelism. The algorithm is not difficult to implement, compared to alternative rendering methods that include various algorithms for specific material or lighting settings.

In addition, the following most crucial disadvantages of the typical light field rendering methods are mitigated by the proposal:

- **Excessive memory requirements**
  Compared to previous methods, which require the entire light field grid to be present in the GPU memory, the proposed method only requires the four closest views to be used in the algorithm; see Section 3.6. Other allocations, such as the focus map image, can even be downscaled without a significant quality loss; see Section 3.4. Also, no pre-trained deep-learning weights are necessary to be uploaded to the GPU memory.

- **Excessive streaming requirements**
  The proposed method is compatible with the proposed compression method based on GPU-accelerated video decoders; see Section 2.7. These decoders proved to be the optimal choice for the streaming of light field data.

- **Focusing artifacts**
  The experimental evaluation of the proposed method showed that a high-quality, all-focused image can be produced as a result; see Sections 3.4 and 3.6. Certain artifacts can still be visible, but the quality of the result outperforms the majority of state-of-the-art approaches. Furthermore, the possible artifacts might not be clearly visible, as the scene might be focused at a distance which is near the correct one or focused somewhere else where the color similarity metric ensures that the part of the image looks natural.

- **Time consuming reconstruction with missing structural information**
  The inputs required for the proposed method are the input images and the aspect ratio of the capturing cameras. The method does not require any depth information as input; see Section 3.2. Other methods use, for example, depth maps, which might not always be available, their estimation would slow down the process, and their usage would increase the memory requirements. No heavyweight 3D reconstructions, data preprocessing, or camera intrinsic parameters are necessary to produce an all-focused view in real time. In addition, no deep learning training process is necessary prior to using the method.

- **Limited view space**
  The proposal focuses on the best quality interpolation of the views in the space enclosed by the input light field grid. Possible extrapolations outside of the grid are beyond the scope of this thesis. The virtual camera can freely move inside the scene bounding box defined by positions of the input cameras.

Experimental measurements revealed that the method outperforms other state-of-the-art ones. In combination with the proposed compression scheme and GPU accelerated algorithms, the method seems to be an optimal choice for a real-time light field rendering. It can be assumed that Hypothesis was proved in the experiments.

The proposed methods were implemented in several experimental applications that were used for the measurements. All applications serve as proof-of-concept which can be used after necessary refactoring in industrial scenarios. All resources are available online[3] and are free to use for other researchers or developers.

[3] fit.vutbr.cz/∼ichlubna/research

Chapter 4
Light Field Applications and 3D Displays

Light field rendering can be used not only with classic 2D displays, but its significance increases when considering 3D displaying multiview devices. Additional research was conducted regarding 3D displays to ensure the best rendering quality of the input light fields.

This chapter describes additional proposed methods used for input light field data processing before presentation on a 3D display. The main proposal of this thesis, described in Chapter 3, can also be used to produce the necessary input views for the 3D displays. This chapter is not directly related to the main scientific hypothesis and its proof, but is related to the topic of light fields and their practical usage with modern technology. Analogous problems with input views located at defined positions and focus artifacts are addressed. This chapter can be viewed as one of the possible applications of light field principles in industry, and additional scientific contributions are proposed.

4.1 3D Displays

This section describes the technology of 3D displays that simulate 3D perception for the user. These devices are tightly related to light fields, as they physically implement their multi-view and optical features. Fig. 4.1 shows various types of 3D display devices.

On classic displays, users can view a 2D projection of a 3D scene. The depth of the scene can be perceived from the motion of the camera [80, 89]. Objects at different depths would have different apparent velocity. The perceived velocity difference is called the motion parallax. Moving objects in the scene can also serve as a depth cue, even without parallax [244, 253]. When objects move closer and farther from the camera, the projected size change indicates the distance of the object. Perspective projection can similarly indicate depth relations in the scene. Textures, transformed by geometry from the original 2D view, are another important depth cue [198]. Other depth cues exist, such as eye accommodation, learned expectations, contextual analysis of the scene, shadows, etc. [200].

Binocular depth cues [71, 182] play an important role in human depth perception. A straightforward way to simulate this perception is to capture two images of the same scene from a slightly different position. These images can be projected separately into each eye, creating the 3D illusion [35]. Binocular parallax creates the 3D feel by a projection of two 2D images that capture the scene from different positions, one for each eye. Monocular motion parallax is based on the occlusion and motion of the objects in the scene according to the position of the users' heads. Several models of devices, called head-mounted displays (HMDs), are commonly used among consumers, usually referred to as virtual reality devices.

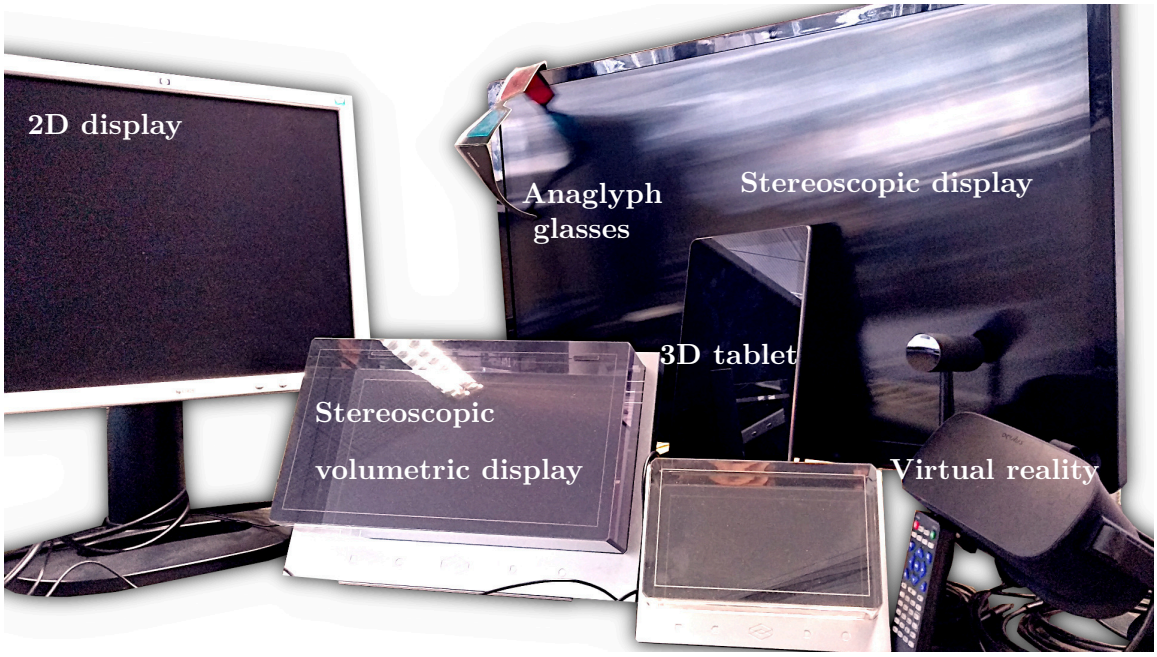


Figure 4.1: Multiple 3D display devices that represent categories from large displays for movie watching, smaller sizes for interactive usage or previews, almost pocket-size devices, and equipment wearable on user's head.

Another way to project different images into each eye is to use light-blocking glasses [56]. The two images can be simultaneously displayed on a screen, each image with different polarization. Polarized glasses on users' heads then pass through only one of the images to each eye. An alternate approach is to use a shutter display. The display cyclically switches between the two images at high frequency. The shutter glasses are synchronized with this switching and pass through the currently projected image only to the associated eye.

Volumetric displays use a rotating mirror and a projector with a high frame rate that is synchronized with the mirror rotation frequency [131]. 3D scene can then be rendered from multiple angles, and each of the views is then reflected by the mirror in the right direction. Users see different parts of the scene depending on their position and viewing angle at the mirror. A more robust solution using a similar principle with a rapidly vertically moving flat screen, on which a different part of the 3D image is projected depending on its current position, are Voxon Photonics products [140].

A different approach that uses the viewing angle-based ray distribution is the Tensor Display [111]. It uses multiple light-attenuating layers that modify the light coming from the source display in the form of directional backlight, which can be simulated by the lenslet array on the display. A compact solution in the form of a 3D tablet by Leia Inc. uses diffractive light field backlighting [78, 142]. Four views of the same scene are projected in 16 directions with a 2D backlight, proprietary optical layer, and an LCD panel. Rapid temporal modulation showing different frames in a short period of time helps to widen the angle of view showing different parts of the scene through the synchronized layers. Similar technology was used in the RED Hydrogen One holographic smartphone [284]. A holographic screen, where each point modifies light rays emitted from the optical module layer in various directions, is used in the HoloVizio display by Holografika [25].

4.1.1 Visual Discomfort

An unnatural stereoscopic input into the human visual system could cause discomfort and eye strain, resulting in excessive brain activity or unpleasant eye conditions such as heterophoria [138]. This discomfort can lead to cybersickness [15]. The effect of varying camera separation, display duration, focal length, and convergence distance on visual perception using stereoscopic displays has already been documented [119, 120]. The eye strain is more severe with increasing disparity between images. The vertical parallax in virtual reality devices is not well tolerated by the human visual system and should not exceed 20″ [189] or 7 mm for two homologous points [285]. An experiment carried out revealed that tilt angle differences of the 3D screen greater than 30° cause noticeable visual discomfort [248]. Similar rules are valid even in the field of augmented reality [147]. Correction of inappropriately shifted or rotated stereoscopic views was proposed and tested by image transformation, revealing the difference between translational and rotational disparities, where rotation was shown to be easier to correct [175]. An objective metric exists to predict visual fatigue from stereoscopic images that measure vertical and horizontal disparities, but considers only two views [69]. The effect of compression was also evaluated, revealing that depth perception does not change significantly with increasing compression ratio, but eye strain is much more significant due to the reduction in image sharpness [236]. However, compression artifacts are less visible in scenes with large disparities [185].

Previously published papers [84, 164, 285] addressed stereoscopic screens and input image distortions with respect to user depth perception. However, novel 3D displays combine stereoscopy with the view-dependent 3D effect in a relatively large viewing cone in front of the screen. The previously discovered rules might not be exactly the same for current state-of-the-art 3D display technology. The issue of distorted capturing camera trajectory is usually not present in already well-explored stereoscopic images because the stereo cameras are usually physically bound together and cannot move independently. Camera distortions can be partially fixed with stereo stabilization algorithms [96, 172]. Parallax of 2° was shown to cause visual discomfort with stereoscopic displays [174]. Another work stated 40′ as the upper limit of the comfort zone parallax [275]. The same paper contained information that young people are more prone to visual sickness than old ones. Research in the field of visual discomfort shows that input views presented on 3D displays need to follow certain rules. Otherwise, the quality of user experience might deteriorate.

4.1.2 Looking Glass

The state-of-the-art multiview display by Looking Glass Factory (LKG) [81, 82] is designed to directly display a 3D scene without any additional equipment, such as glasses, similarly to lenticular displays [255]. This device consists of a classic 2D display, a microlens layer, and a glass block on top. USB output is also included, since the device calibration with the display and microlense parameters is sent to the rendering computer to ensure the correct pixel alignment and optimal 3D projection.

Multiple views are displayed at once on the underlying display in a defined pattern; see Fig. 4.2. The microlenses redirect the rays per pixel in the right directions, so that it is possible to see different views according to the user's position. The glass layer refracts the rays coming out of the background display so that they reach wider viewing angles. One of the limitations of this display is only horizontal parallax, where users are expected to be sitting in front of the screen and tilting their heads from side to side. This device can be considered as a hardware implementation of the one-directional discrete light field

approximation. Similarly to 1D light field rendering, an array of images, taken along a horizontal line with constant spaces between the camera positions, is used as an input; see Fig. 4.3. The display renders the appropriate view based on the user's viewing angle and also blends neighboring views to mask out abrupt changes of the views.

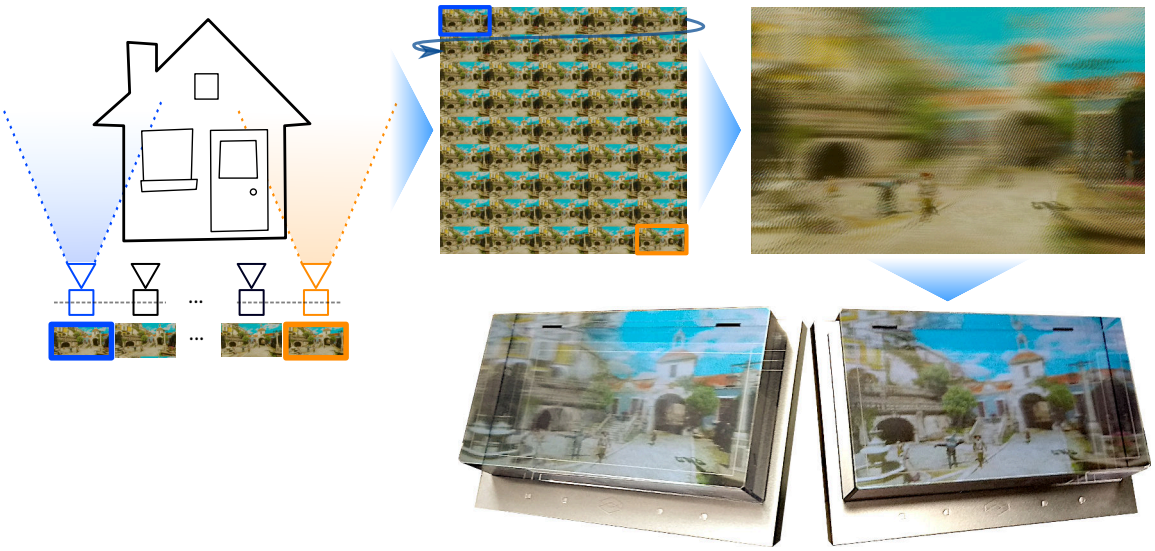


Figure 4.2: Evenly distributed cameras on a horizontal line capture the scene as a set of images that are merged into one matrix (quilt) representing a 1D light field. The top left image is the first one from the linear sequence and the bottom right the last one. The quilt is then transformed into the internal LKG format which can be directly displayed on LKG. The optical block on the device distributes the pixels in the correct directions.

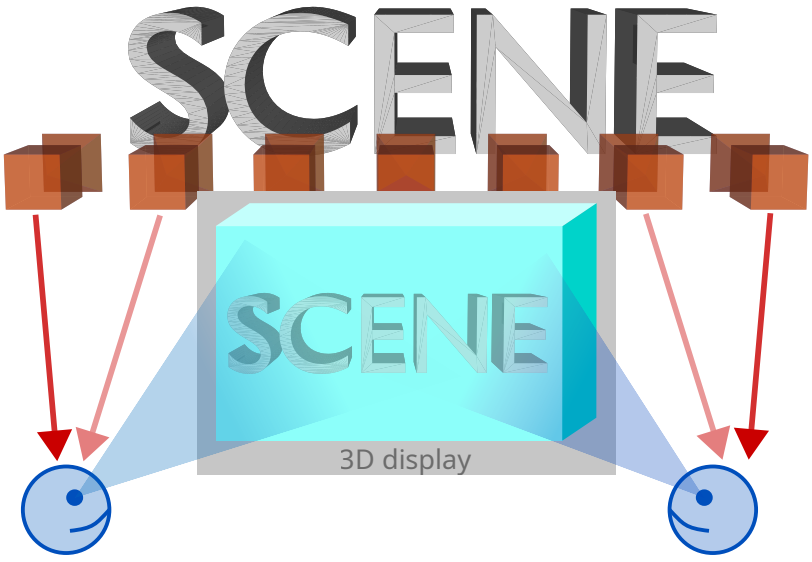
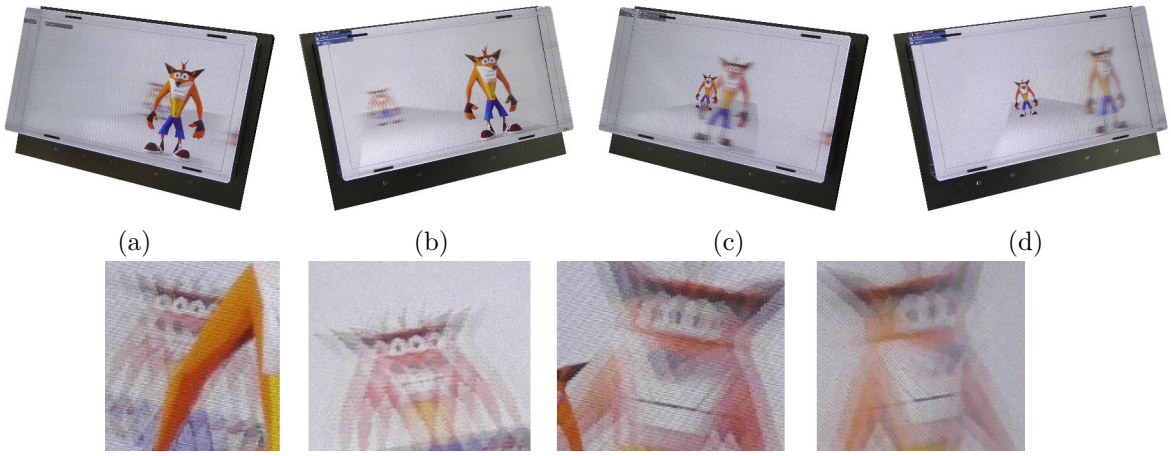


Figure 4.3: The figure shows the optimal orientation of the capturing cameras which are placed in the scene according to the horizontal 3D display orientation. Users then see a combination of the captured views according to their viewing position.

Focusing

Focusing of the scene is closely related to a similar issue in light field rendering [158]. A distance from the camera's near plane where all projected pixels are in focus is called the zero-parallax plane. Such pixels are spatially static and do not show any parallax motion when users change their viewing angle. Their color can change due to reflections or other lighting effects. The parallax is visible in the defocused parts of the scene and is one of the important depth cues that simulate the resulting 3D effect [218]. An example of different focus distances and the parallax effect on the physical device is visible in Fig. 4.4. The geometric representation of the focus problem is shown in Fig. 4.5.



(e) Parts of defocused areas from each view are zoomed.

Figure 4.4: The figure contains a 15.6″ LKG device with the same scene, captured from the left (a,c) and the right (b,d) side and with zero-parallax plane positioned at the front (a,b) and back (c,d) object. The parallax is visible when the view angle changes. The out-of-focus blur resembles ghosting artifacts due to the mixing of the rendered views.

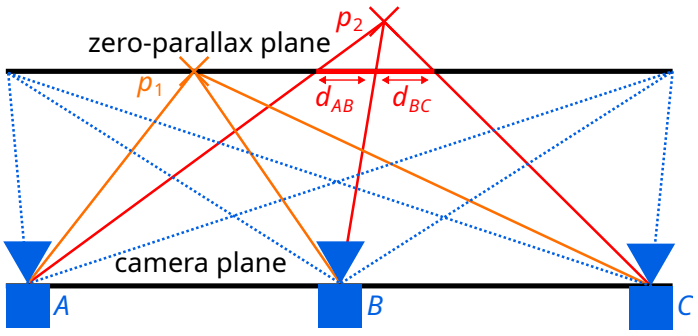


Figure 4.5: Point p_1 lies on the zero-parallax plane and is focused from all cameras A, B, C. Point p_2 lies further away and is projected on the screen with disparities d_{AB}, d_{BC} between the views.

Focusing is simulated by displacement of the input images according to their position on the capturing trajectory. The method is similar to the shift-sum algorithm [13] used in light field rendering [123]; see Fig. 4.6. Light field function L describes light from a given angular $\mathbf{u}$ and spatial $\mathbf{x}$ coordinates with the refocusing parameter $focus$; see Eq. 4.1.

$$focused = \int_{\mathbb{R}} \mathrm{L}(\mathbf{x} - \mathbf{u} \cdot focus, \mathbf{u}) \mathrm{d}\mathbf{u} \tag{4.1}$$

After shifting, the last pixel is repeated at the edge of the image to avoid visual discontinuities. The $focused$ result, where the ith image of the total N images of the scene is shifted ($\mathrm{shiftImage}_i(amount)$) by the $focus$ value, which can be relative or in pixels, can be calculated by Eq. 4.2. The summation can be a specific pixel distribution operation, for example, conversion into the internal format of the LKG. The images can be moved to the left or to the right, while the center image remains static.

$$focused = \sum_{i=0}^{N} \mathrm{shiftImage}_i(focus \cdot (1 - \frac{2i}{N})) \tag{4.2}$$

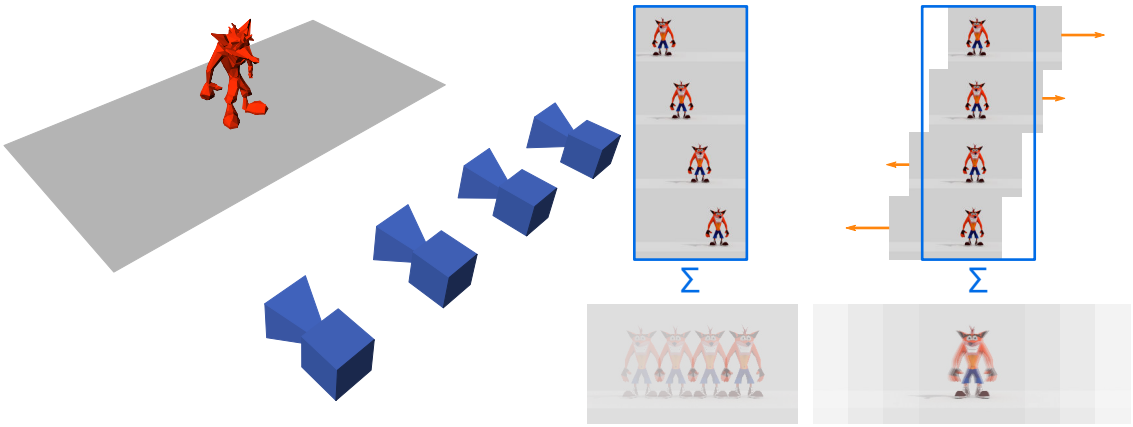


Figure 4.6: The scene is captured from multiple positions. The images can be mixed together as they are, leading to ghosting artifacts. The zero-parallax plane can be changed by shifting the images so that the desired part of the scene is in focus (constant screen position) but does not show parallax.

The combination of the images from the quilt into the internal format is performed according to Eq. 4.3, where h is the index of the RGB channel ($R = 0, G = 1, B = 2$), I function samples an input input image at the given coordinates, and the internal format image access function I_{lkg} is extended by the parameter to index the specific channel. The calibration parameters, defined by the LKG manufacturer, are as follows: s is subpixel shift, t is tilt, p is pitch, v is view portion, and c is center shift. The total number of views is defined as N, $\lfloor\rfloor$ means floor operation, and $\{\}$ is the extraction of fractional part.

$$\mathrm{I}_{lkg}(x, y, h) = \mathrm{I}(\lfloor\{(x + sh + yt)p - c\}\, N\rfloor, vx, vy) \tag{4.3}$$

Each physical LKG device comes with a different set of these parameters. The equation allows to directly render the quilt on the display without any proprietary software as it distributes the pixels according to the structure of the microlens array on LKG. The focusing can be achieved by the transformation of x, y coordinates similarly to Eq. 2.4.

The user always sees multiple mixed views at the same time. Their disparity leads to ghosting artifacts in parts far from the zero-parallax plane. Scenes with a large depth range would show more parallax and 3D feel, but they would contain more ghosting artifacts in the defocused parts. A naive approach to create an optimal all-focused result would be to

split the image into parts, where each part would be focused with a different zero-parallax plane position. This principle is similar to the per-pixel focusing method from Section 3.2. However, this would cause discontinuities when changing viewing angles and an unnatural motion of the objects that would rotate instead of moving to the sides according to the parallax; see Fig. 4.7.

(a) Left view shows abrupt splitting of the scene and duplicated objects.

(b) Right view shows a discontinuity on the ground.

Figure 4.7: Two zero-parallax planes (different focus distance for each half of the image) were used to keep both objects focused. The parallax is visible and is sharply cut at the middle of the image. The motion of the objects is also unnatural. Both are rotating instead of one seemingly moving to the sides.

4.1.3 Camera Trajectory of Input Views

Input views for 3D displays need to follow defined rules regarding their camera positions. Methods mentioned in this section are tools used to identify camera motion in a sequence of such views. These methods are used in the following sections.

Camera motion following a straight horizontal line trajectory is called *truck* in cinematography. It is visually similar to *pan* which consists only of rotation around vertical axis. Dense [77] or sparse [177] optical flow can be used to detect camera motion for consecutive video frames. Another option is to use feature matching algorithms with descriptors such as SIFT [19], SURF [30], KAZE [14], or ORB [222] and find the displacement of the found features.

For a sparse optical flow, the tracking features are selected using the Shi-Tomasi corner detector [126]. The dominant camera motion in a video sequence is usually estimated by averaging optical flow vectors in predefined regions of the image/flow [294]. A set of rules is then used to estimate camera motion based on the dominant direction in each region [153, 288].

Knowing the intrinsics of the camera, a combination of multiple optical flow models can provide more accurate results [205]. A spatio-temporal derivative of the intensity in two successive frames can be used [37]. Motion vectors can be computed using temporal gradient-based block matching. The resulting motion is estimated using the motion vectors of interest, which are chosen from the field according to their significance and consistency [105]. If encoded video files are used as input data, the necessary motion vectors can be extracted from the compressed stream [16, 260, 282]. Hidden Markov Models [193] or Transferable Belief Model [95] might be more robust alternatives to solid thresholds in the camera motion estimation rules. Multiresolution least-squares methods might be used to

fit the motion model even with noisy data [199]. Model-based estimations might not cover the whole range of possible videos and camera motions. Machine learning approaches to overcome this problem were also proposed [50, 51, 86, 170, 173], but their quality depends on the training process.

Structure-from-motion and visual simultaneous localization and mapping techniques are frequent approaches that are used to reconstruct a point cloud of a 3D scene or to retrieve the camera trajectory from a sequence of images. ORB_SLAM3 [47] is based on a previous SLAM research [191, 192]. ORB_SLAM3 approach consists of tracking, mapping, relocalization, and loop closing based on ORB features extracted from the image frames. Camera pose information can be retrieved in real time for each frame after the initialization phase when the scene map is created. The quality of the result, aside from the quality of the input sequence, depends on the camera intrinsics, especially the focal length value.

4.2 3D-Display-Friendly Frames Extraction

Suitable views of scenes for the 3D display can be captured intentionally, but many shots can be found, for example, in movies or gameplay videos. The method proposed in this section [4] can automatically extract suitable views of a scene from any video footage or detect that no such scene exists. This proposal describes light field capturing from existing data. It is important to ensure that such light field is suitable for the 3D display so that the main proposal of this thesis can also be applied in this field.

With minor adjustments, the method could be used to assist users during a capture process on a set. Without the aid of the method, the user would need to use special equipment to ensure the optimal motion of the camera along the desired trajectory. In case of already recorded videos, manual, time-consuming selection of the video frames would be necessary with additional editing and repeated evaluation on the display. Automatically extracted sequential frames, the so-called *quilt*, are ready to be displayed on a 3D display. LKG was used as a testing device. This method proposes the entire pipeline to produce a visually appealing result. The input video is analyzed, suitable frames are extracted, resampled, and the optimal focus distance is identified to reduce out-of-focus artifacts; see Fig. 4.8.

4.2.1 View Synthesis Approach

One possibility of obtaining the views for LKG is to pick a few frames from a video, reconstruct the scene, and render the missing views. The 3D reconstruction fails in most cases because the scene is captured only from one direction in the input views. 3D reconstruction algorithms usually expect the scene to be captured from many different positions. The suitable sequences for LKG do not contain views positioned freely in the space. For example, the expected LKG-friendly *Barber* scene from the original dataset was used in INGP [190], FILM [217], Meshroom, and 3DF Zephyr. The scene was reconstructed in the two programs, but the result is not optimal, as shown in Fig. 4.9. INGP pre-processing of the data by Colmap failed. The FILM deep learning frame interpolation produced a frame between the boundary frames from the video. However, the frames had to be downscaled from FullHD to HD due to insufficient resources on NVIDIA® RTX™ 2070. The interpolated frames contained visible artifacts even when the distance between the views was

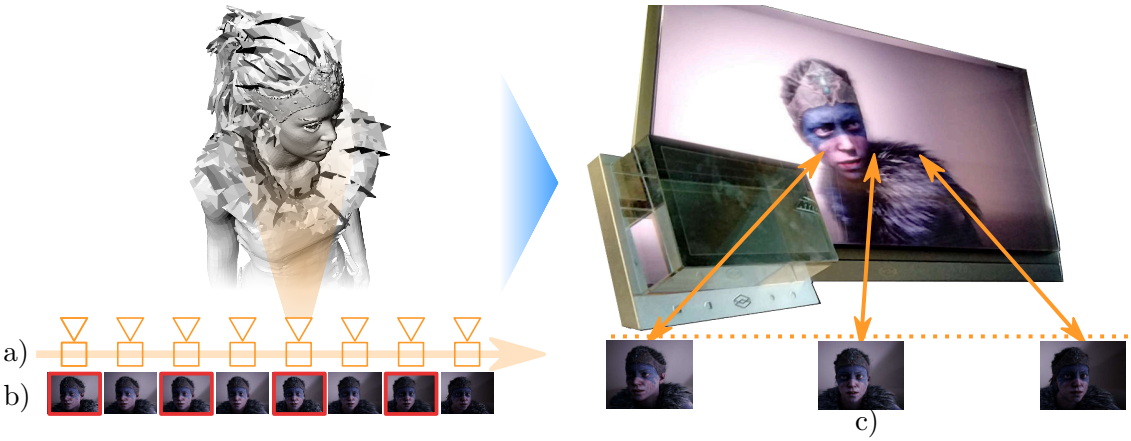


Figure 4.8: The figure is an overview of the proposed method: a) The identification of video frames captured by horizontally moving camera. b) Selection of desired images for the sequence. c) The conversion of desired images into proper format which can be displayed on the LKG. Different virtual views are visible from different real-life viewing angles.

halved. RGBD photos can also be processed by LKG software and displayed, but the lack of parallax and missing reflections create the unwanted cardboard effect [291].

The only reliable way to produce a good-quality quilt from videos is the proposed method with possible interpolation on a small scale if the views are not distributed linearly along the trajectory. The proposed focusing metrics are still necessary for the extracted or interpolated views. The user can capture the scene in a similar way as when taking a panorama photo with a mobile phone. The only difference is that instead of rotating the phone, the user would move to the side. Such captured footage would be automatically prepared by the proposed method for direct rendering on the 3D display.

4.2.2 Extraction Overview

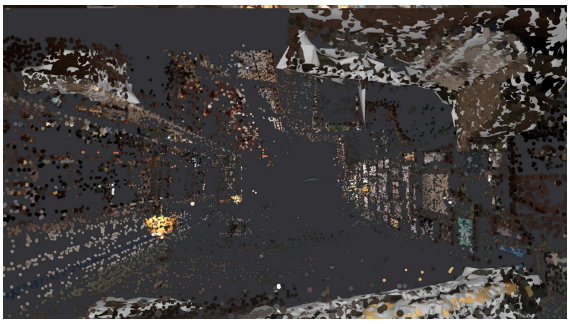
The proposed method consists of the following steps:

1. **Quilt extraction** where the input video stream is analyzed and suitable frames are chosen for further processing. The suitable frames are those that capture the scene having their camera positions lying on a horizontal trajectory.

2. **Frame resampling** where The camera in the selected sequence can be subject of acceleration or uneven capturing frequency producing uneven spacing between the views. Equally-distanced frames are selected from the sequence to ensure the best visual quality of the rendered result.

3. **Automatic focusing** solving the problem of 3D displays support for only one distance with a fairly limited depth of field, where the scene is visually sharp and focused. The optimal focus distance is estimated to ensure that most of the scene is in focus.

Sparse optical flow, dense optical flow, feature matching, and SLAM approaches were used to estimate the camera trajectory and compared in the reference implementation. In the first phase, a rough camera motion estimation is performed, detecting truck and pan motion sequences. Pure truck sequences are ideal for LKG. The camera can also do a combination of truck and pan, which might still be acceptable. Excessive panning motion reduces the parallax effect. The second phase determines the amount of pan motion in the



(a) Meshroom reconstructed only standalone polygons.

(b) 3DF Zephyr produced a distorted but full geometry.

(c) Original middle view was taken from the input light field.

(d) FILM produced a distorted middle view.

(e) Original quarter view was taken from the input light field.

(f) FILM produced a distorted quarter view with less artifacts.

Figure 4.9: Zephyr produced the best result when reconstructing the scene, but the result still contains a lot of artifacts. The deep FILM interpolation produced distorted results for both full and halved distance between the views from *Barber* scene in the used dataset.

sequence. Additional checks are performed, such as motion blur and shakiness detection. A score that evaluates the suitability of the sequence is the result of the second phase. An approach using ORB_SLAM3 does both tasks in one phase.

4.2.3 Horizontal Sequence Detection

Pairs of pixel blocks are identified as belonging to the same spot in the scene in two consecutive video frames. The field of motion vectors (optical flow) is then known for each frame pair. The mean of these vectors is a good guide to determine the overall motion of the camera. Monocular visual SLAM methods also exploit a similar concept of feature matching in two frames, and based on them, the camera pose or scene map is estimated. Fig. 4.10 shows a simple scenario in which the ideal horizontal quilt sequence is mapped to a subset of suitable video frames.

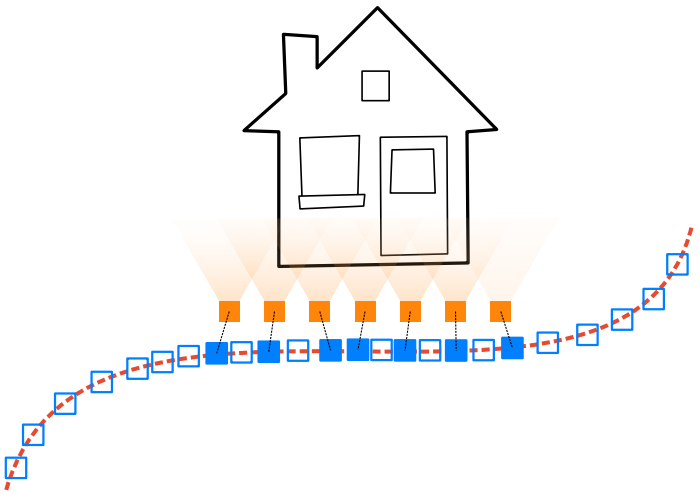


Figure 4.10: The camera positions on the trajectory belong to frames in the input video. The frames sample the camera motion with a constant framerate. The required ideal horizontal sequence, capturing the scene from a desired spot and with a given spacing, is then mapped on the closest input frames.

A possible scene change can be detected by a simple histogram comparison or by setting a threshold for the estimated camera pose change. The result of the analysis is an array of estimated camera pose differences between frames. The analysis phase is the most computationally expensive part of the processing.

In many cases, the mean value of these vectors can be used directly to decide whether the truck/pan motion is dominant in the shot. When the truck motion is combined with pan, the parallax can be inverted, depending on the distance of the objects in the scene from the camera. To solve this issue, the direction of the sequence can be ignored, working only with the magnitude of horizontal motion. The method accepts the ith frame f_i in the sequence seq_i until the vertical velocity vel_{vert} is above a defined threshold t_{vert} or the horizontal velocity vel_{hor} is close to zero; see Eq. 4.4 and Alg. 7.

$$seq_i = \begin{cases} accept & \text{if } \text{vel}_{hor}(f_i, f_{i-1}) > 0 \wedge \text{vel}_{vert}(f_i, f_{i-1}) \leq t_{vert} \\ reject & \text{otherwise} \end{cases} \tag{4.4}$$

Data: Video frames sequence fr
Result: Clips suitable for LKG quilt *clips*
$motions = \text{emptyArray}()$;
foreach *two consecutive frames prev and next* $\in fr$ **do**
    $motions \xleftarrow{\text{insert}} \text{motion}(prev, next)$;
$clips = \text{emptyArray}()$;
$clips \xleftarrow{\text{insert}} \text{emptyClip}()$;
foreach *id and motion* $\in motions$ **do**
    $\{x, y\} = |motion_{xy}|$;
    if $x > x_{min}$ ***and*** $y < y_{max}$ **then**
        $\text{last}(clips) \xleftarrow{\text{insert}} id$;
    else
        if $\text{hasSufficientLength}(\text{last}(clips))$ **then**
            $clips \xleftarrow{\text{insert}} \text{emptyClip}()$;
        else
            $\text{last}(clips) = \text{emptyClip}()$;

Algorithm 7: The algorithm iterates over the calculated camera offsets between all consecutive frame pairs and is adding frames into the sequence as long as the motion meets the requirements. Note that in SLAM-based approaches, where the camera rotation is explicitly known, the maximal rotational bounds and checks can be implemented in addition to the vertical velocity test.

4.2.4 Pan Elimination

Fig. 4.11 shows images composed of multiple video frames. The pan shots are very similar to the truck ones, except for the distortion in the corners of the field. The angles between the motion vectors and the horizontal axis are higher than zero. Dense optical flow can be computed for each corner of the image pair. The mean motion vector should point down on one side and up on the other side of the image or vice versa, according to the camera motion direction. Each fifth pair is tested in the reference implementation for performance reasons.

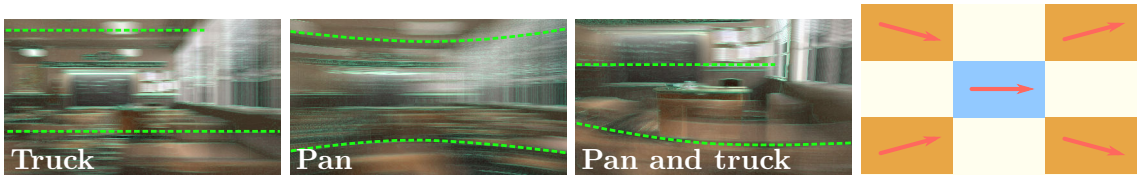


Figure 4.11: The apparent motion of the pixels is demonstrated in the images for truck, pan, and combined camera motion. Frames from the testing sequences were mixed together and the image gradient of the result was highlighted. The rightmost image shows how pan could be identified from the motion field based on the previous images.

A score is calculated for each processed sequence; see Eq. 4.5. The calculation of the score s takes into account the overall amount of motion *blur* in the sequence, *shake*

(amount of vertical motion), the average presence of the *pan* pattern, and the difference *dist* between the average amount of vertical motion at the center of the image y_{cent} and at the corners y_{cor} normalized by the maximal allowed limit $maxDiff$; see Eq. 4.6. The amount of blur is measured using the variance of Laplacian method [206, 210]. Lower-scored sequences are assumed to be more suitable for LKG. The weights w_i can be adjusted according to the specific requirements of the quilt. In the measurements, the weights were set to $0.2, 0.5, 0.1, 0.2$ in this order. The values were determined according to a previous study on user experience related to camera trajectory distortions on LKG [6]. It shows that the pan pattern can negatively affect the result quickly, while shaking can be tolerated to a certain extent.

$$s = w_1 \cdot dist + w_2 \cdot pan + w_3 \cdot shake + w_4 \cdot blur \tag{4.5}$$

$$dist = \min\left(1, \frac{y_{cor} - y_{cent}}{maxDiff}\right) \tag{4.6}$$

In the case of ORB_SLAM3 analysis, the camera pose matrix is decomposed into translation and rotation. The translation difference can be obtained simply by subtracting the translation vectors. The rotation difference $diff_{rot}$ is estimated by comparing the rotational quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ according to Eq. 4.7. Pure pan motion is not suitable for SLAM reconstruction and is rejected automatically.

$$diff_{rot} = 1 - \left|\frac{2\text{acos}(\mathbf{q}_1 \cdot \mathbf{q}_2)}{\pi} - 1\right| \tag{4.7}$$

4.2.5 Non-linear Camera Velocity and Noise

The motion vectors are available from the analysis phase and can be used to improve the quality of the accepted sequence. The vertical component of the average motion vector can be used to reduce the shaking of the camera on the vertical axis. The camera motion perpendicular to the viewing plane (*dolly* motion) can be used to change the scale of the images to reduce the noise on the depth axis.

For the best viewing experience on LKG, equally distanced views of the scene are optimal [6]. This requires a constant camera velocity and frame rate, which is not guaranteed in the input. If the frames are not equally spatially distanced, the sequence has to be sampled non-uniformly. The frame distance can be represented by a horizontal component of the average motion vector between frames. Horizontal position changes are accumulated in *acc* frame-by-frame, with an increasing index i. The nearest neighbor can be accepted as the nth frame in the resampled sequence res_n when the accumulated motion exceeds the maximal allowed value m_{max}. The difference is labeled as d in Eq. 4.8. The conditions that decide the index of the accepted frame that is closer to the desired position are described in Eq. 4.9 and Alg. 8.

$$d = acc - m_{max} \tag{4.8}$$

$$res_n = \begin{cases} i & \text{if } d < |d - |m_{i-1}|| \\ i - 1 & \text{otherwise} \end{cases} \tag{4.9}$$

Data: X axis motion values for a sequence of selected frames $clip_x$
Result: Sequence of frame indices with linearized motion $clip_{lin}$
$motion_{max} = \max(clip_x)$;
$acc = |\text{first}(clip_x)|$;
$clip_{lin} = \text{emptyArray}()$;
foreach *two consecutive id and motion* $\in clip_x$ **do**
    $acc = acc + |motion_{next}|$;
    if $acc \geq motion_{max}$ **then**
        $delta = acc - motion_{max}$;
        if $delta < |delta - |motion_{prev}||$ **then**
            $clip_{lin} \xleftarrow{\text{insert}} id_{next}$;
        else
            $clip_{lin} \xleftarrow{\text{insert}} id_{prev}$;
        $acc = acc - motion_{max}$;

Algorithm 8: The sequence is resampled in the algorithm to ensure a constant camera position difference between the frames. Note that instead of choosing the closer frame according to the velocity accumulator, a frame interpolation can be used to create a new synthetic one.

The distances between frames have to be large enough to create sufficient 3D perception. If the frames are too close, the LKG result looks flat without significant parallax. On the other hand, too large distance reduces the amount of focused area in the scene. The sequence can be resampled according to the optimal frame distance; see Fig. 4.12. The computation of the best alignment offset o_{best} from the range (o_s, o_e) of the detected sequence is described in Eq. 4.10. The $\mathbf{s_{optimal}}$ is a vector that contains the positions of the views in the optimal sequence with constant spacing. The $\mathbf{s_{detected}}$ is a vector that contains the positions of the views in the sequence that was detected in the previous phase. The optimal sequence is shifted by the offset o and the error between the sequences is computed, looking for its minimum. The vector and scalar addition is defined as the addition of the scalar value to all values in the vector.

$$o_{best} = \underset{o \in [o_s, o_e]}{\arg\min} \{ |(\mathbf{s_{optimal}} + o) - \mathbf{s_{detected}}| \} \tag{4.10}$$

Eq. 4.10 is implemented in practice by Alg. 9 that describes a quilt window with the desired distance. This window is sliding over the extracted frames, and the error is computed from the distance between the actual frames and the positions in the window. The window position with the lowest error marks the best sequence according to the given requirement. The optimal distance depends on the content of the scene and the user preferences.

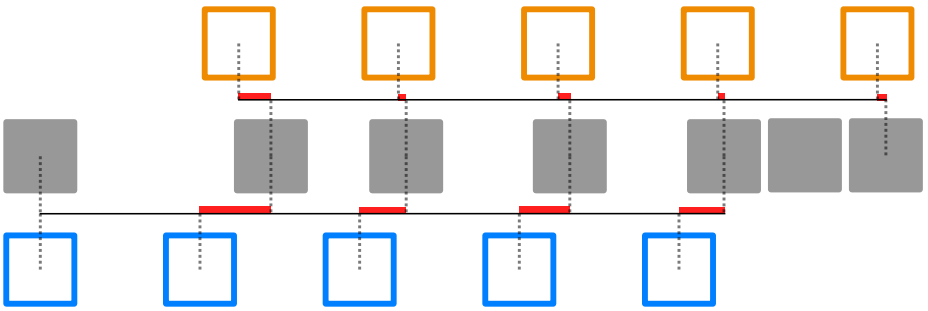


Figure 4.12: The middle row depicts the video frames spaced according to their camera displacement. A quilt window with fixed spacing is placed at the beginning in the bottom row and at a better position in the top row. The distances between the positions in the sampling windows and the positions of the nearest input frames are defining the overall error.

Data: Sequence of frame positions pos, desired spacing between the views $space$, location of interest in the input sequence loc, searching distance around the defined position $dist$, $resolution$ of the search, number of the quilt views $quiltSize$

Result: An offset of the window with desired parameters with the lowest error $clip_{best}$

$clips = \text{emptyArray}()$;
$step = \frac{2 \cdot dist}{resolution}$;
$range = (loc - dist, loc + dist)$;
foreach $offset \in range$ ***with*** $step$ **do**
    $error = 0$;
    $window = \text{emptyArray}()$;
    foreach $\{i \in \mathbb{Z} \mid 0 \leq i < quiltSize\}$ **do**
        $sample = offset + i \cdot space$;
        $nearest = \text{findNearest}(pos, sample)$;
        $diff = |nearest_{pos} - sample|$;
        $error = error + diff^2$;
        $window \xleftarrow{\text{insert}} nearest_{id}$;
    `// if duplicates are not allowed`
    if $\text{hasNoDuplicates}(window)$ **then**
        $clip = \text{newClip}(error, offset)$;
        $clips \xleftarrow{\text{insert}} clip$;
$clip_{best} = \text{lowestErrorClip}(clips)$;

Algorithm 9: A quilt window is sliding along the sequence, computing the squared distance between the closest samples and predefined positions in the window. The window position with the lowest error is then marked as the optimal one.

4.2.6 Focusing

The display blends multiple frames to avoid discrete frame changes when users change viewing position. Scene content positioned at *zero-parallax plane* is always sharp and maintains its screen-space coordinates; see Fig. 4.13. The further the content is from the plane, the more parallax is present along with the out-of-focus artifacts. The zero-parallax plane position can be changed by shifting the views so that the same area of the scene lies on the same screen-space coordinates.

Figure 4.13: The first picture shows the 3D scene with two zero parallax plane positions. The two other pictures are the actual results displayed on the LKG showing how the plane position affects the focusing in the rendered image.

The positions of the start and end points of the acceptable focus range, where at least parts of the scene are focused, depend on the depth range of the captured scene; see Fig. 4.14. The proposed method finds the optimal zero-parallax plane position that makes most of the scene focused.

The unique internal image format, displayed directly on the LKG screen, where all quilt views are combined into one image, is analyzed. The areas of the image outside the zero-parallax plane are distorted by a diagonal blur-like pattern. The more the pattern is visible, the less focused the given part of the image is on LKG. Two focus estimation metrics are proposed.

The first is a difference of Gaussians (DoG) edge detection on the internal LKG format image. The second performs a subtraction of two internal LKG images with different display calibration settings. The diagonal blur pattern is different for each calibration. In both cases, the out-of-focus pixels in the result contain high values. A similar principle was used in all-focused light field research [148]. Fig. 4.15 shows both the metrics and their results.

The algorithm iterates over a wide focus range and searches for the minimum of the given metric, as shown in Fig. 4.16. The principle is the opposite of the standard blur measurement methods [73]. The Gaussian metric is described by Eq. 4.11 and subtraction by Eq. 4.12. The optimal focus distance f_o^{DoG} and f_o^{sub} searched as f in the interval (f_s, f_e) is computed, for the first metric, as a difference of internal LKG format images I_{lkg} with calibration parameter c_1, processed by Gaussian filter G with filter parameters p_1 and p_2. The second metric subtracts two different internal images with calibration parameters c_1 and c_2. The DoG parameters used in the measurement were $radius = 10, \sigma_1 = 0, \sigma_2 = 10$, but different reasonable parameters showed the same results. The calibration parameters used in Eq. 4.3, c_1 were set to $t = -0.1153, p = 354.42108, c = 0.04239, v = 0.99976$ according to a physical device used in the tests, and the parameters c_2 were changed to $t = 0.05, p = 150$ which showed the most visible changes after subtraction.

$$f_o^{DoG} = \underset{f \in [f_s, f_e]}{\arg\min} \{G(p_1, \mathrm{I}_{lkg}(c_1, f)) - \mathrm{G}(p_2, \mathrm{I}_{lkg}(c_1, f))\} \tag{4.11}$$

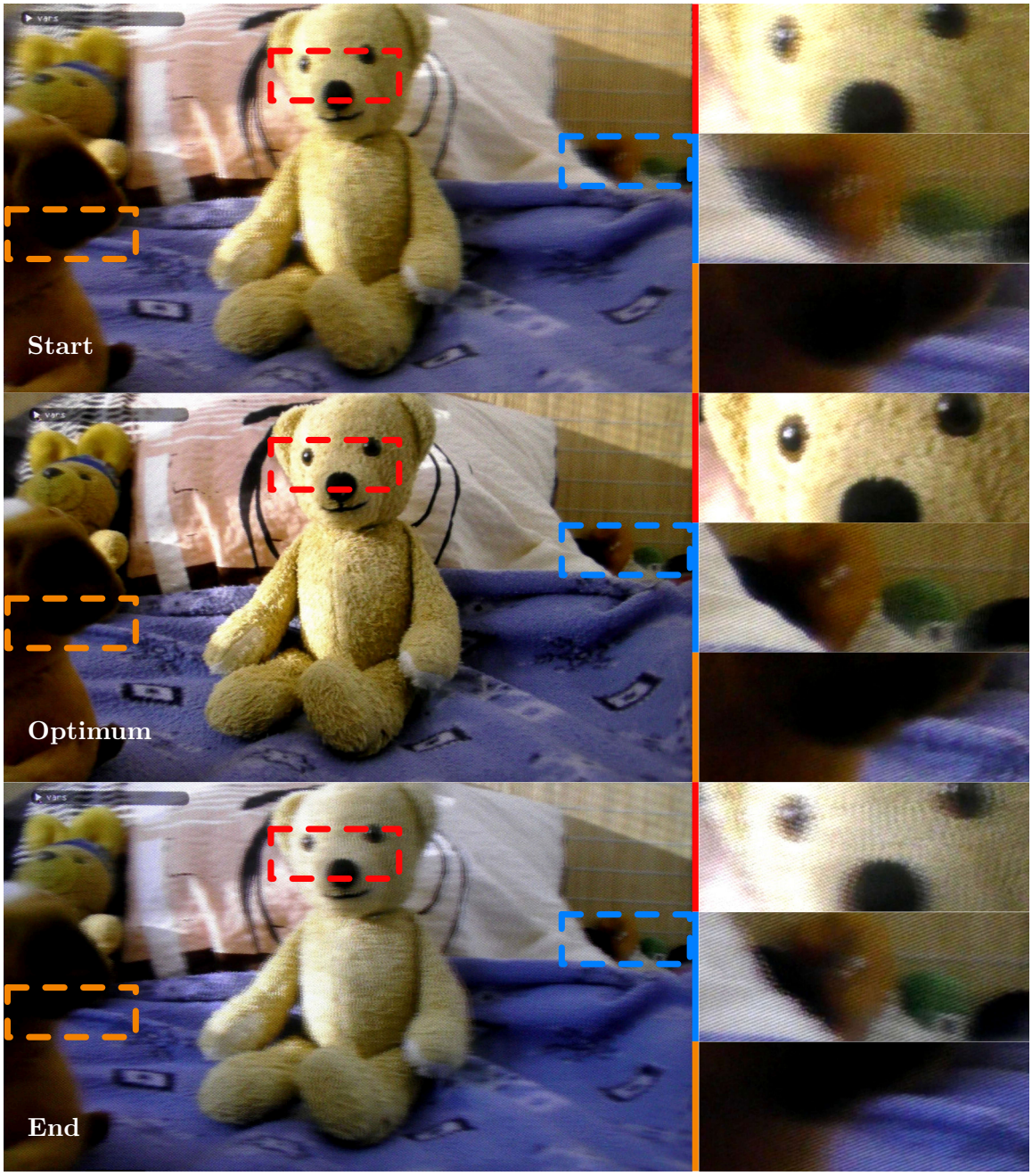


Figure 4.14: Scene is focused at the borders of the focus range and in the optimal distance where most of the scene appears to be focused.

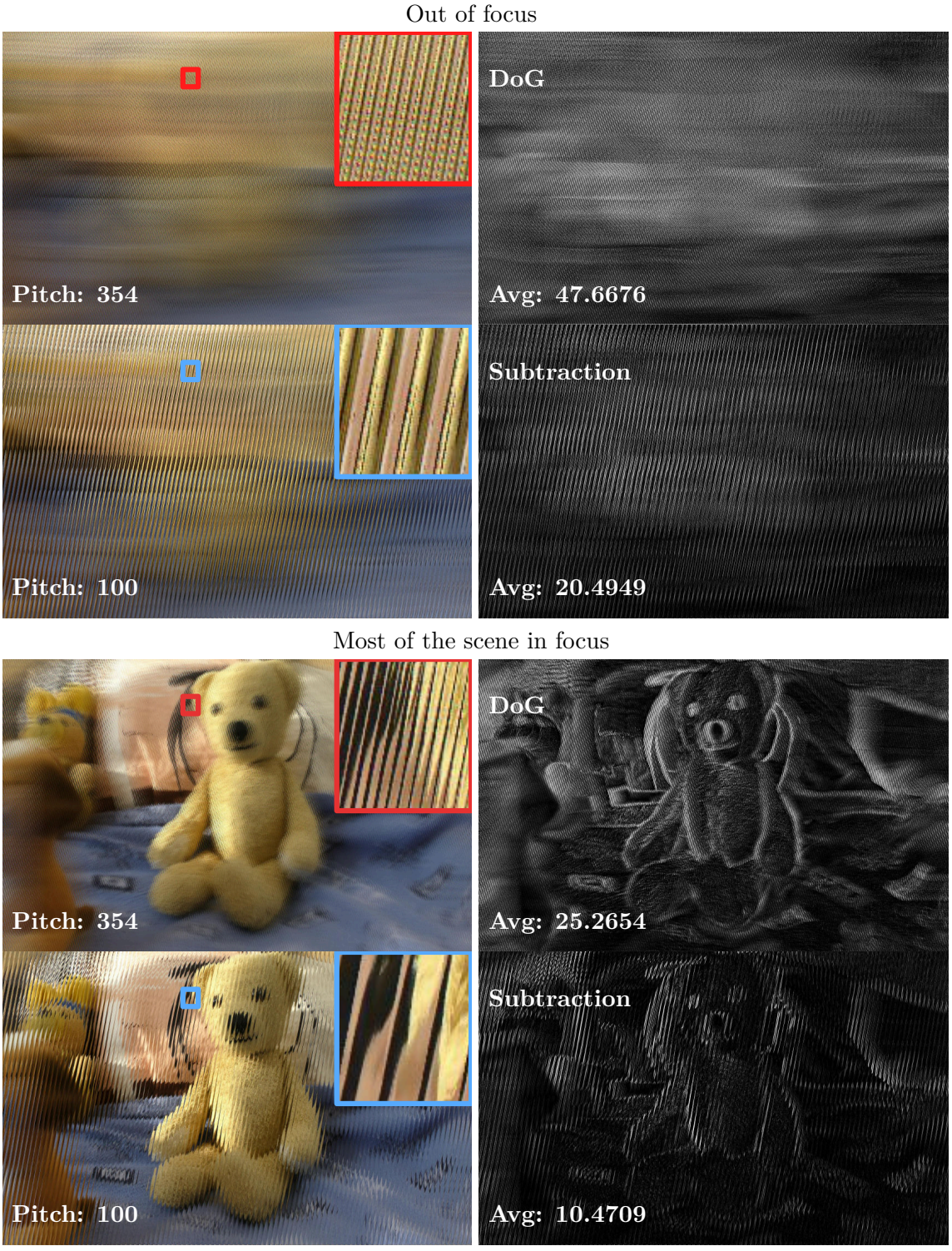


Figure 4.15: Focused and out-of-focus images are shown. Left ones are in the LKG internal format with two different display calibration settings. The first proposed metric is the difference of Gaussians from a single LKG image, and the second one is a subtraction of the two images with different calibrations. The more focused the image, the less energy (right images) the metrics yield.

$$f_o^{sub} = \underset{f \in [f_s, f_e]}{\arg\min} \{ \mathrm{I}_{lkg}(c_1, f) - \mathrm{I}_{lkg}(c_2, f) \} \tag{4.12}$$

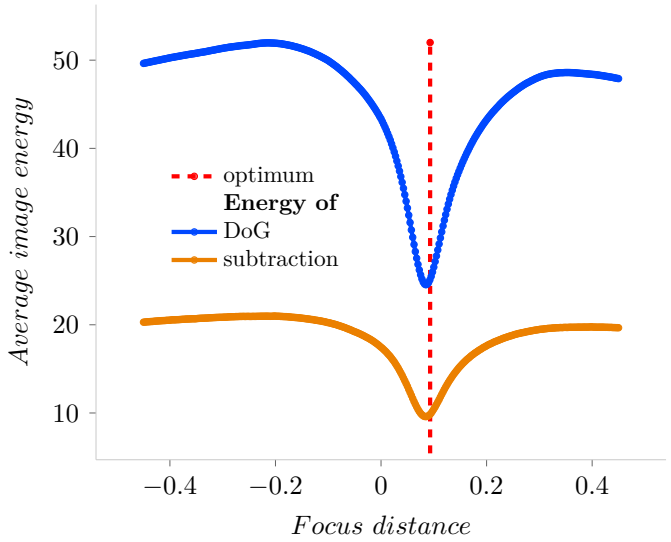


Figure 4.16: Two focusing metrics are plotted with manually marked optimum distance where most of the image is focused. The location of the global minimum of the energy marks the best focus.

4.2.7 Results & Discussion

The overall quality of the detector was evaluated in terms of efficiency and performance, see Tab. 4.1, in the following experiments:

1. **Quilt extraction** which aim was to measure the accuracy of automatic quilt extraction from a video. The classification is evaluated using an annotated dataset.

2. **Frame resampling** experiment which proves that frames can be sampled in an optimal way, resulting in a lower error compared to a simple first frame alignment.

3. **Automatic focusing** of the resulting quilt proves that the optimal focus distance where most of the scene lies on the zero-parallax plane can be found automatically, which makes the resulting quilt ready to be displayed.

OpenCV implementations of the Lucas-Kanade, Farneback, and ORB methods were used along with the original ORB_SLAM3 implementation for the quilt extraction. All experiments were executed on a machine equipped with NVIDIA® GeForce RTX™ 2070 GPU and Intel® Core™ i5-8500 CPU 3.00 GHz CPU, running Arch Linux.

The dataset consists of 182 FullHD, 25 fps, synthetic and handheld camera scenes encoded in H.265 format. The scenes are divided into 7 categories according to the dominant camera motion: chaotic, dolly, pan, pan and truck combined, pedestal, tilt, and truck. The duration of the videos is from 2 to 13 seconds. The videos were captured to cover all types of scenes with various depths, close and far objects of interest, etc. The synthetic shots were rendered in Blender with pre-made scenes from Blender demo files. They were captured in

Horizontal sequence extraction

analysis method	true [%]	false [%]
ORB_SLAM3	79.7	20.3
ORB	75.6	24.4

Optimal window positioning

scene	squared error	
	first frame aligned	optimized
teddy	382.988	251.594
hut	502.925	300.505
class	115.437	82.4649
pavilion	88.3094	72.9947

focus distance detection

method	absolute error	
	DoG	subtraction
average	0.0475	0.0441
min	0	0
max	0.549	0.549

Table 4.1: The table contains important results from the experiments with the best results highlighted. ORB_SLAM3 and ORB showed the best results for quilt extraction. True values are considered $TN+TP$ and false $FN+FP$ (T/F is true/false, P/N is positive/negative) according to the annotated dataset. The alignment errors of the first frame of the quilt window and optimal placement are compared, proving that the proposed algorithm can find a better position in the sequence.

two versions: with ideal smooth camera motion and with additional shaking and motion blur. The real-life videos were captured by Panasonic HC-VX980 Camcorder without any special equipment.

4.2.8 Quilt Extraction

The desired point in Fig. 4.17 marks the maximal vertical motion between two frames that is accepted as valid. The position was empirically determined by watching the amount of artifacts on LKG. An experiment was conducted in which identical frames with a simple pattern were vertically shifted with an increasing offset. The first kind of artifact is caused by the mixing of the frames, which occurs throughout the displaying range to simulate the binocular parallax. Fig. 4.18 shows the difference between the vertically displaced quilt and the aligned one. These artifacts start to occur with displacement around 1.5 px. The second type is visible in between two frame positions in the monocular motion parallax when watching the display from a close distance. This kind of artifact is more sensitive to displacement and was detected starting at 0.5 px displacement. A value between these two, 1 px, was selected as desired; see Fig. 4.19.

Three metrics, the Youden index, closest to point $(0, 1)$, and maximum area, were used to identify the best ratio threshold in the ROC curves [67], which corresponds to a vertical

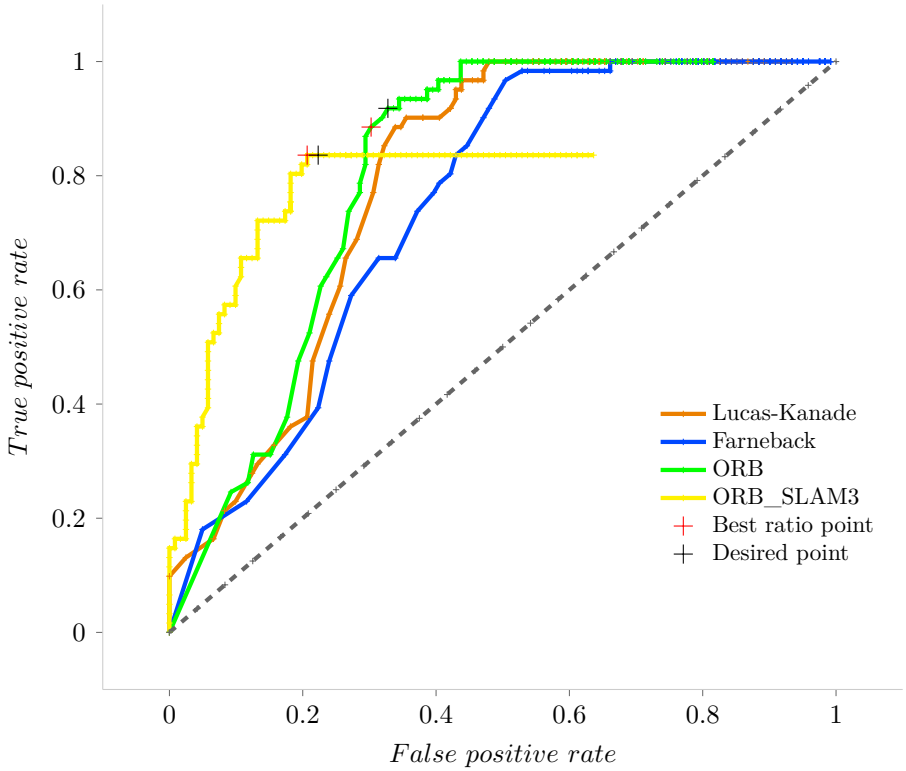


Figure 4.17: The ORB_SLAM3 and ORB are the best candidates for quilt detection. The threshold parameter is the limit of vertical motion. ORB_SLAM3 does not cover the whole ROC range due to its inability to analyze all the sequences. The desired point (1 px vertical motion tolerance) was chosen according to empirically found maximal acceptable threshold and the best ratio (0.9 px tolerance) according to the commonly used metrics.

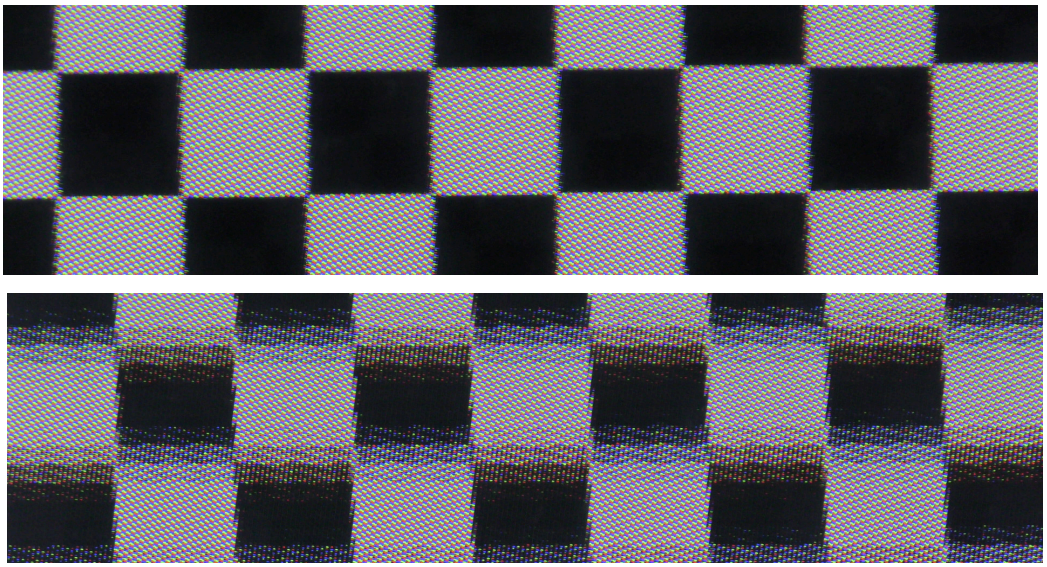


Figure 4.18: The first picture shows a checkerboard quilt with views that are aligned in one horizontal line. The second picture shows the same quilt with vertical displacement between views where artifacts are visible.

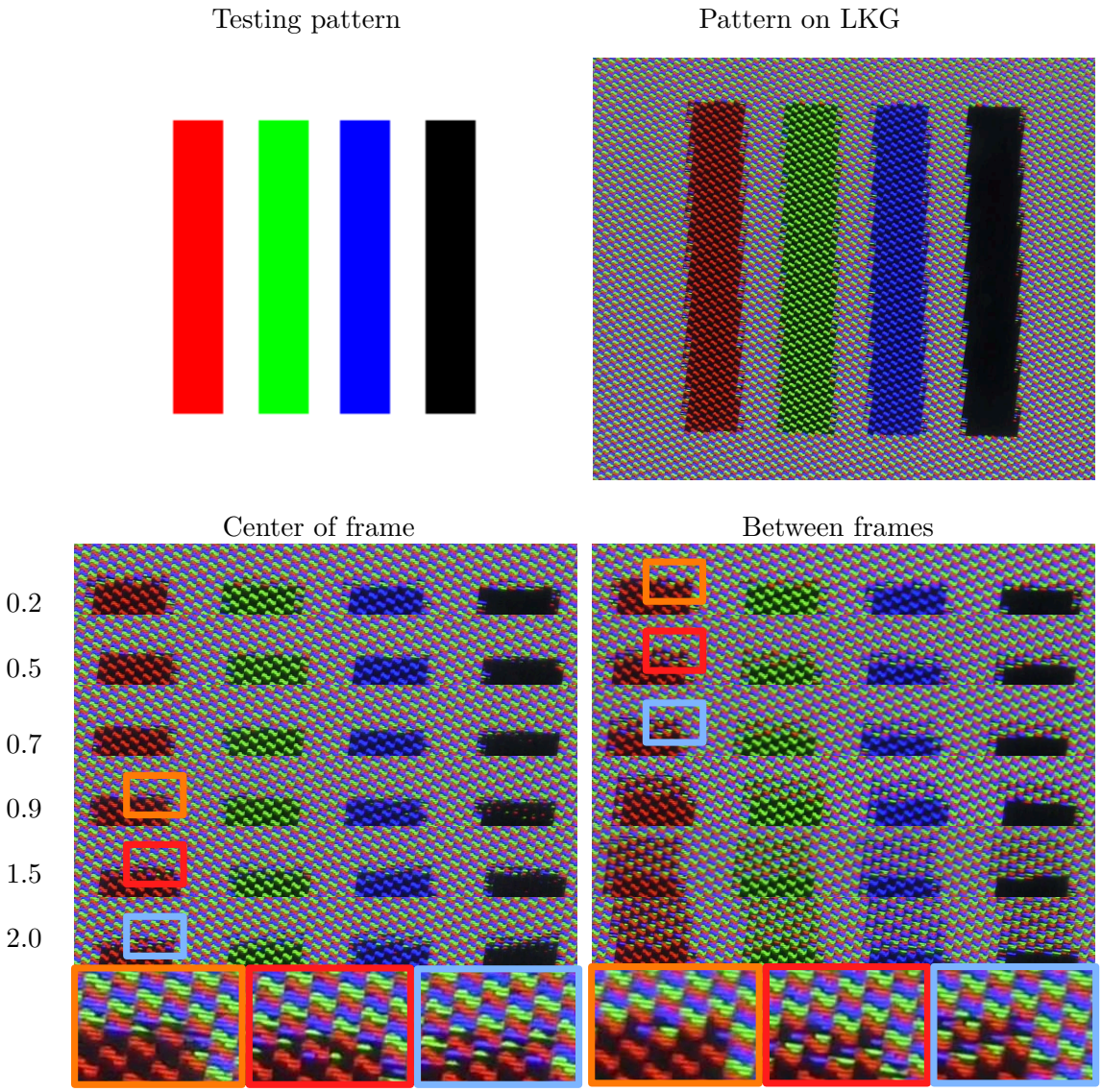


Figure 4.19: A simple pattern was chosen to measure how vertical motion limit affects the quality of the result. The border of the pattern is getting distorted with the increasing vertical displacement (values on the left). The breaking points were detected at 1.5 px in the center case and 0.5 px in the in-between case.

motion tolerance of 0.9 px. Fig. 4.20 shows the distribution of negative and positive results among the categories in the dataset. Fig. 4.21 shows the accuracy and precision results.

ORB-based feature detection method outperforms the optical flow approach. ORB_SLAM3 shows better results than all other approaches, but it is limited by its inability to analyze all possible sequences. Most of the problematic sequences are true negatives, but some positive sequences were also rejected. That is the reason why its ROC curve does not reach the right top corner as the rest.

An unknown focal length is assumed in the measurements, according to the use case, where the user can provide a random video without further information. The focal length was estimated by Meshroom photogrammetry software, as it is a necessary parameter for ORB_SLAM3 method.

The results of the computational time measurements are presented in Fig. 4.22. ORB might be more versatile and robust, while ORB_SLAM3 is more accurate in classification. However, the efficiency of ORB_SLAM3, depends on the focal length estimation, which might be problematic in scenes shot by multiple different cameras or shots containing zooming.

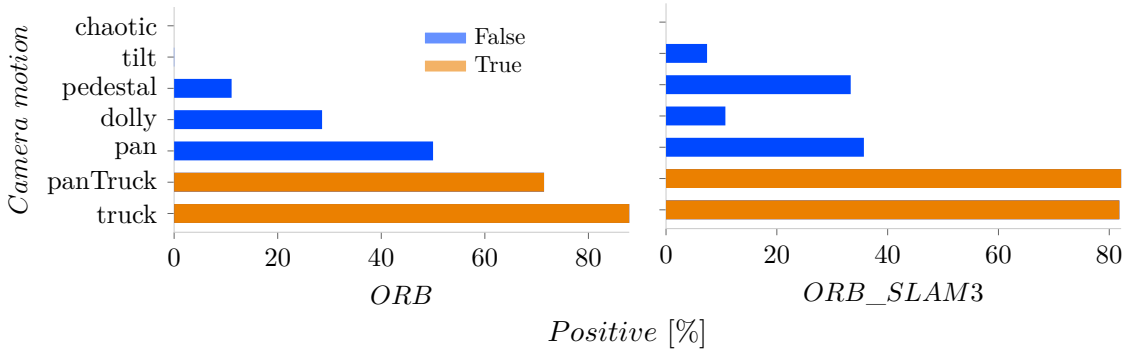


Figure 4.20: The amount of positive ORB and ORB_SLAM3 results in each category of the dataset is depicted. Only pure truck or truck-and-pan combined camera motions are true positives. Truck has the most positive results, followed by pan and truck, which can still be displayed but are prone to artifacts and a flat look.

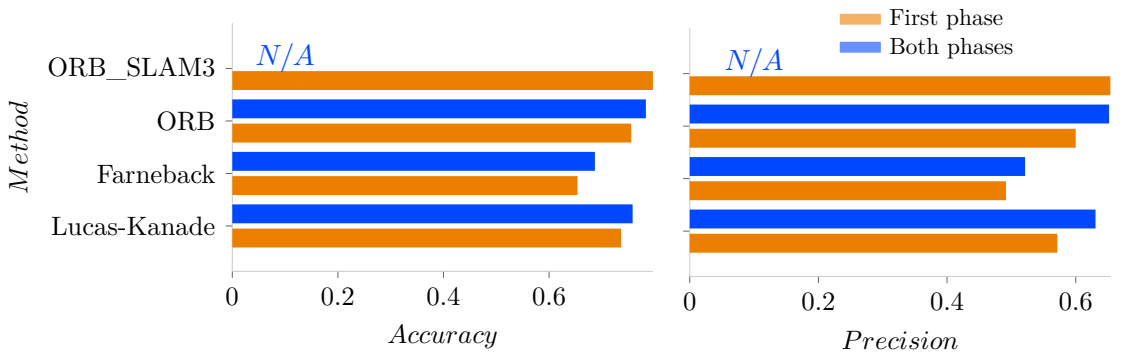


Figure 4.21: The figure shows precision and accuracy values for all methods. The vertical motion limit is set to 1 px. ORB_SLAM3 method is the best choice for the analysis.

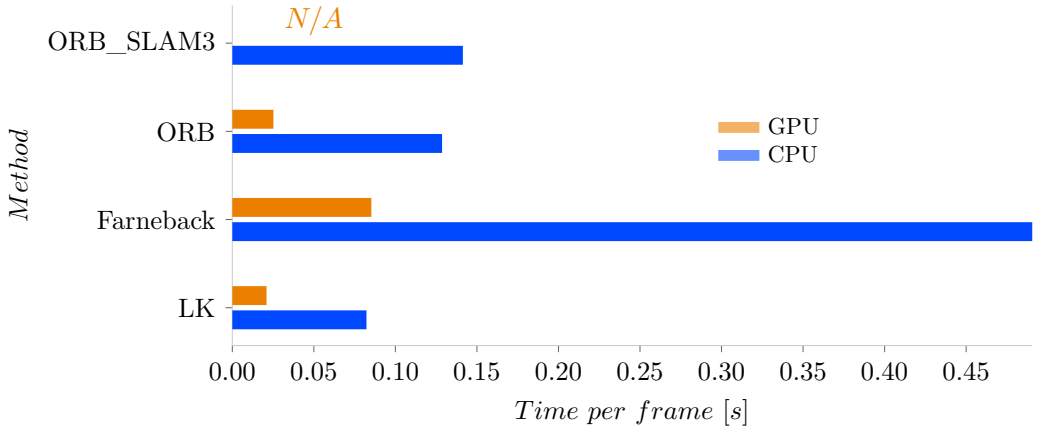


Figure 4.22: The chart compares the computation times of both CPU and GPU implementations of the analysis methods. Times were measured and averaged over the whole dataset. The resolution of the processed sequences is 1920×1080.

4.2.9 Frame Resampling

teddy sequence from the *truck* category in the dataset was chosen because it is recorded with a handheld camera and is bound to have different spaces between the frames. Fig. 4.23 shows how the resampling algorithm selects the optimal offset within the defined range to minimize error.

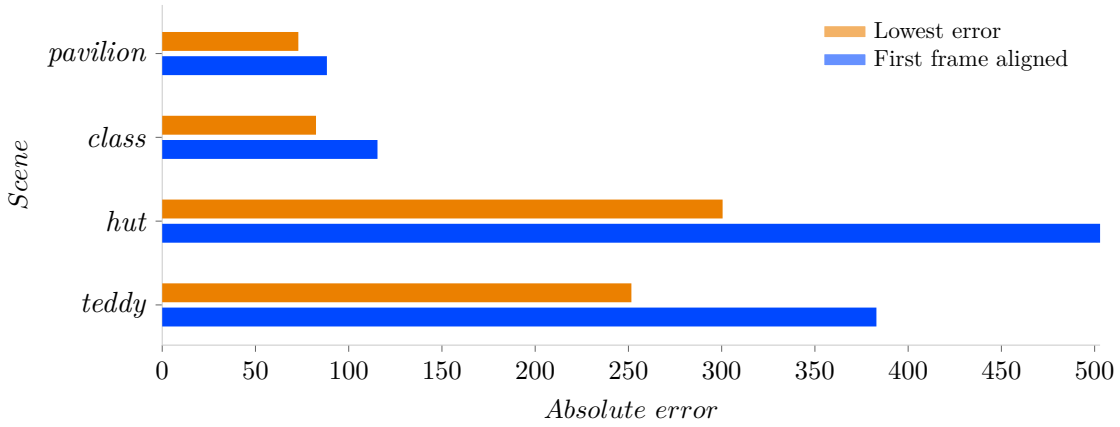


Figure 4.23: The chart shows the results of the experiment with the adjustment of the position of the quilt window in scenes from the *truck* category. The distance between the frames was twice the average distance in the sequence to avoid duplicates. The search range was set to one quarter of the distance.

4.2.10 Automatic Focusing

The focusing metrics were evaluated on the quilts created from the truck sequences. The range and optimal value were manually annotated, evaluating the result on LKG. The absolute error of the estimated optimal value and the annotated one was calculated and normalized by the whole scanning range. SMAPE was used as a second evaluation metric. The results of the two proposed focusing metrics do not differ significantly, as shown in Fig. 4.24. The optimal focus distance is in all quilts very close to the metric minimum.

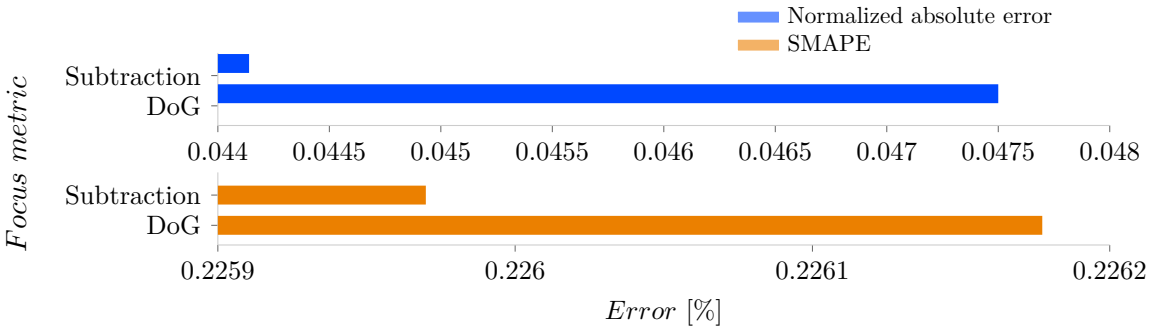


Figure 4.24: Bars are showing the errors of both focus estimation metrics. The differences are not significant.

4.2.11 Comparison to Existing Methods

To prove the novelty of the proposal, the method was compared experimentally with similar methods. No alternative full-quilt detection methods for 3D displays were proposed before. Only the subtasks of the proposal were compared in standalone tests. The results show that the proposal cannot be replaced by existing standard approaches.

Quilt Extraction

3D reconstruction software, such as Meshroom, can be used to estimate camera poses, but it takes minutes to process one frame compared to near real-time processing proposed in this paper. It would also be problematic to process a video file without knowing how many frames are necessary to include in the computation. The proposal is focused on processing of the two subsequent frames, retaining the context of the previous results. In most cases, two frames are not enough for a successful 3D reconstruction.

Panorama stitching software is another option as it detects the pan motion with possible shaking when the capturing device is handheld. Experiments with the Autostitch program [40] showed that pure truck motion shots can be properly detected as a suitable sequence and processed by the software. However, the combination of pan and truck does not yield acceptable results, and this category of potentially suitable quilts would be rejected. Simple averaging of motion vectors from optical flow would also make the distinction between small amount of camera vertical shake, pan, and truck impossible.

Research in video analysis [214] usually focuses only on the general classification of the shot type such as *moving* or *static* and these approaches cannot be used for the quilt detection. CAMHID [105] method can be used to classify camera motion, but the results presented by the authors mention only static, pan, tilt, and zoom categories without the necessary distinction between pan and truck, which is necessary for the proposed

method. Similar categories translation (pan and/or tilt), zoom, and static are defined in other works [64, 95, 276].

Truck motion is distinguished from pan in a novel deep learning approach of camera motion detection for story and multimedia information convergence [22] where 8 frame intervals are processed. An advanced motion vector extractor preprocesses the shots, and ResNet is used to classify the resulting sequences.

The available implementation of the network was trained with the same data that the authors provided. The network was then used to measure its accuracy on the data tested in this paper to determine if the network can be used instead of the proposal. The network can classify a shot into pan, pedestal, tilt, truck, zoom-in, and zoom-out categories. This classification is not detailed enough for the quilt detection because truck with a small amount of pan is also acceptable. This motion is not distinguishable by the network. The measurement was slightly adjusted in favor of the network approach, and any shot from the combined truck and pan category classified by the network as either standalone pan or truck is considered to be successful true positive, and false classification between other categories is not penalized. Tab. 4.2 shows that the proposal in this paper is more suitable for the task than the alternative deep learning network.

analysis method	**true** [%]	**false** [%]
proposed	79.7	20.3
deep [22]	68.1	46.9

Table 4.2: The table compares alternative existing detection method with the proposal. True values are considered $TN+TP$ and false $FN+FP$ according to the custom annotated dataset.

The frames containing the motion vectors need to be downsampled to resolution of 600×300 compared to 1920×1080 in this paper's proposal. The time of the classification for one frame in the network is 0.006 seconds and the time of optical flow preprocessing is 0.1 seconds compared to 0.025 seconds for ORB and 0.14 seconds for ORB_SLAM3 in the proposal of this paper. The proposal is not a real-time solution and both compared methods can be labeled as the same speed category since the difference is not significant. If the same resolution frames were used as in the proposal, the deep classification time would be at least 0.07 seconds and the preprocessing would be 1.15 seconds due to the processing of 11.52× more data.

Focusing

To prove that a novel focusing metric is necessary for the quilt detection task, an attempt to identify focus distance with three other methods was conducted. The first is a standard contrast-based method [240] with root mean square (RMS) contrast metric [149]. The second is a no-reference image quality metric LIQE [296] which should be able to detect various artifacts and defocused areas. Both metrics were used on the internal LKG format like in the proposal and on the blended views according to Eq. 4.2 with all weights set to the same constant, resulting in averaging of the refocused images. The third is a PSNR comparison of the refocused views according to the central one. Each view is shifted according to the focus distance, compared with the central one with the PSNR metric, and the average of the PSNR values is taken as the metric. The results are shown in Fig. 4.25.

The maximal values of both metrics on the internal LKG format are close to the optimum but not as close as proposed metrics in Fig. 4.16. The error of the proposed metrics on *teddy* scene is only 15 px in the focus displacement while the contrast metric error is 36 px which is more than 2× higher than in the proposal. The charts of the proposed metrics also show steeper curve towards the minimum which identifies the area of potential other focus distances that the user might want to choose. The slope of the alternative methods is either noisy or not that steep so such identification would be more problematic. The idea of evaluation of the internal format is proposed in this paper, so the internal versions of the metrics are only semi-alternative. The versions with blending, which can be considered as an alternative approach, are not suitable at all. The conventional methods cannot be used in this case.

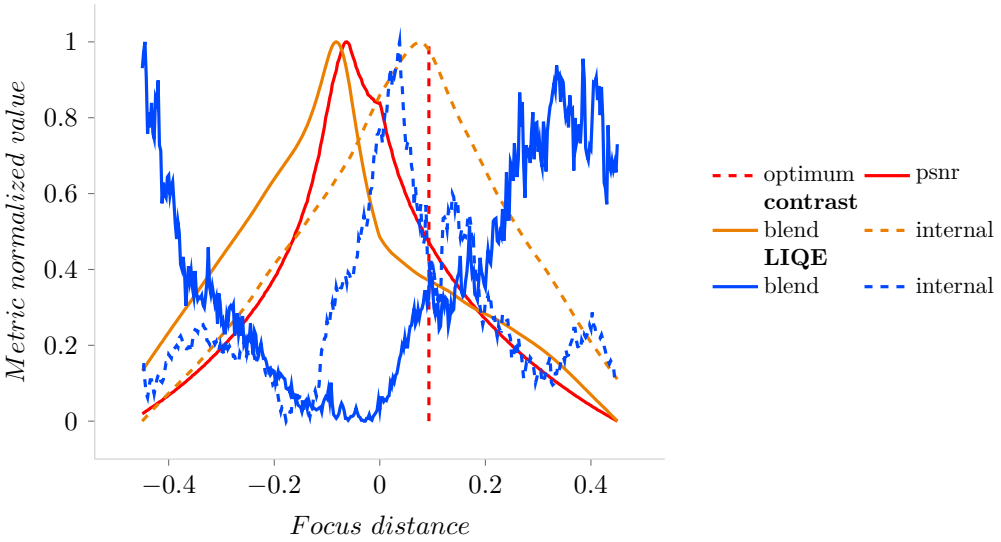


Figure 4.25: Alternative existing metrics that can be used to measure focus in image were compared on *teddy* scene. The values were transformed to the range of (0,1) because only possible minimum or maximum are the desired information. The results show that these metrics are not suitable for 3D displays. The metrics were tested on internal LKG format and blended views.

4.3 Focusing Artifacts Mitigation

Another study was conducted to improve the quality of the quilts displayed on LKG [3]. The focusing on 3D displays and possible transformations of the input scenes are necessary to ensure the best viewing comfort for the user. The light field rendering method proposed in this thesis is capable of producing the input views for 3D displays but does not address the necessary adjustments of such views for optimal viewing on the devices. This section addresses this issue.

The novel saliency-based autofocusing method was compared to the previously proposed one and proved to be more efficient. Additionaly, a data prefiltering step is proposed to improve the existing method. Low-pass filtering to reduce interperspective aliasing was implemented as a depth-of-field blur to mitigate out-of-focus artifacts. The measurements showed that such filtering, proposed in previous works, can improve the result but might not be optimal in all cases. The theoretical basis and the results of the experiment concerning the relation of the distance between objects in the scene and the focus settings are documented. A novel scene-warping method was implemented to increase the focus areas in the result and was proved to be efficient and desirable by users.

4.3.1 Edge Detection Focusing

The previously described focusing method, based on DoG is extended. The output values of the metric for each focusing distance need to be prefiletered prior to the minimal value identification. When the scene is extremely defocused, the total energy of the DoG image tends to get very low; see Fig. 4.26. This happens at the borders of the focus range; see Fig. 4.27. The global minimum would not mark the correct focus. Therefore, the average energy value is calculated and all energy values from the borders of the array are removed until they are higher than the average. The whole process is described in Alg. 10.

4.3.2 Saliency Based Focusing

This method is content-aware. First, the potential object of interest is detected, assuming that the user would like to see this object focused [36]. The goal of the method is to position the zero-parallax plane near the detected object to minimize visual discomfort [132]. To identify the object of interest, the deep learning method GICD [297] is used. For each image of the input set, a saliency map is computed and saved. The map contains white pixels in the area of interest and black elsewhere. The method iterates over a focus range and shifts the maps according to the zero-parallax plane position. The intersection of the shifted maps is then computed; see Fig. 4.28. The intersection is implemented by median filtering across the images to avoid completely black saliency maps clearing the result. The focus distance with the highest energy saliency map intersection is then the one which has the object of interest in focus; see Alg. 11.

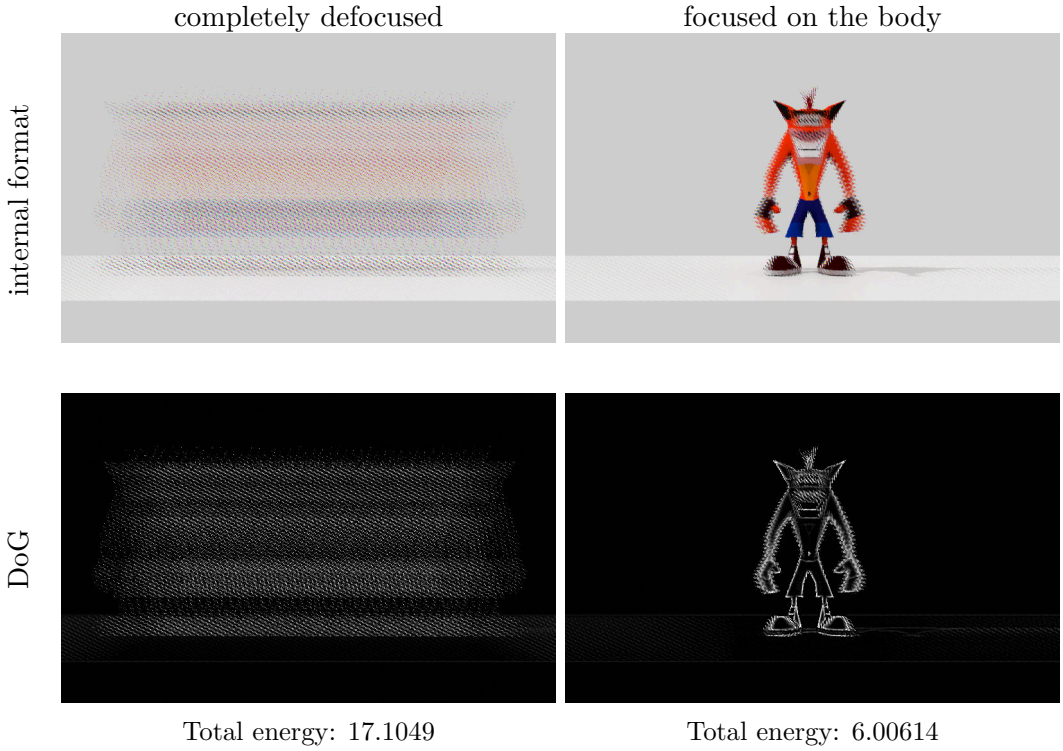


Figure 4.26: The first row contains the internal LKG format images with two different zero-parallax plane positions. The second row shows the same images after DoG edge detection. The resulting energy of the image is lower in the focused case.

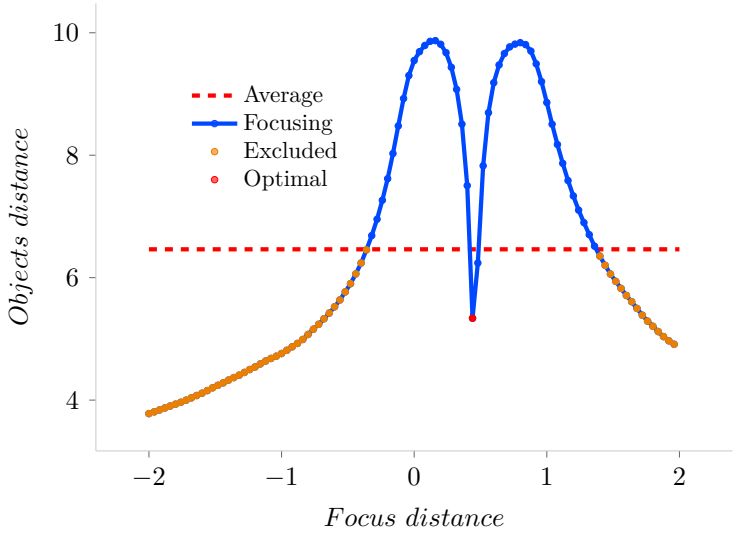


Figure 4.27: The minimal energy value of the entire scan range would not be correct due to the very low values at the boundaries of the range. Therefore, the border values are excluded until they reach the average value. The values are taken from the evaluation of the *Buddha* dataset.

Data: The input set of $images$, discrete $focusRange$ with defined step
Result: Optimal focus $distance$
$values = \text{emptyArray}()$;
foreach $focus \in focusRange$ **do**
    $interal = \text{generateInternal}(images, focus)$;
    $dog = \text{differenceOfGaussians}(internal)$;
    $e = \text{energy}(dog)$;
    $values \xleftarrow{\text{insert}} \{focus, e\}$
$avg = \text{getAvg}(values)$;
foreach $v \in values$ **do**
    if $v_{energy} > avg$ **then**
        break;
    $\text{setAsInvalid}(v)$;
foreach $v \in \text{reverse}(values)$ **do**
    if $v_{energy} > avg$ **then**
        break;
    $\text{setAsInvalid}(v)$;
$optimal = \text{minEnergy}(values)$;
$distance = optimal_{focus}$;

Algorithm 10: The method iterates over a focus range and returns a distance, where the DoG of the internal LKG format has the lowest energy.

(a) Saliency maps are computed from three different views.

(b) Completely defocused maps are mixed. (c) Maps are focused on the body and mixed.

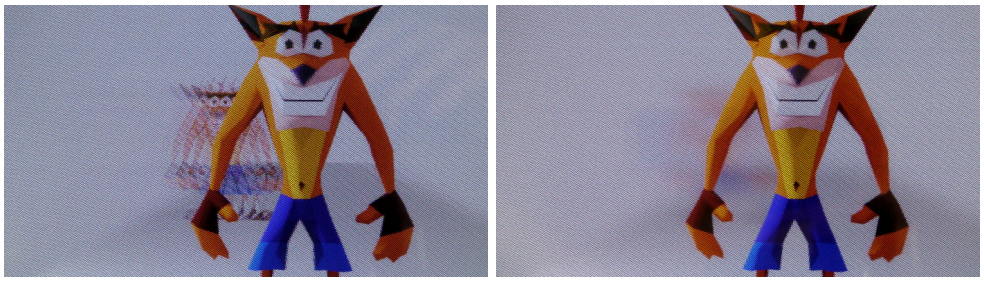
Figure 4.28: The first row contains three generated saliency maps. The second row shows two images which are results of a simple mix blending of the maps to demonstrate, that the defocused case would result in completely black multiplication of the images while the focused one would contain white areas.

Data: The input set of *images*, discrete $focusRange$ with defined step
Result: Optimal focus *distance*
$maps = \text{saliencyMaps}(images)$;
$optimal = \text{minimalValue}()$;
foreach $focus \in focusRange$ **do**
    $maps = \text{shiftImages}(maps, focus)$;
    $mix = \text{multiply}(maps)$;
    $e = \text{energy}(mix)$;
    if $e > optimal \rightarrow e$ **then**
        $optimal = \{e, focus\}$;
$distance = optimal \rightarrow focus$;

Algorithm 11: The method iterates over a focus range and returns a distance, where the intersection of saliency maps has the highest energy.

4.3.3 Depth of Field Enhancement

Due to the discrete light field representation and optical mixing of the views, a ghosting artifact appears in the parallax area. This artifact is similar to defocusing in a human visual system when the eyes are fixed at a certain distance [79]. The images can be pre-filtered to mitigate this interperspective aliasing [186, 301]. Prefiltering removes high frequencies in the scene that are breaking the bounds defined by the Nyquist-Shannon Theorem. Depth-of-field (DoF) simulation blurs the expected defocused areas [151]; see Fig. 4.29. The depth map or information on the geometry of the scene is usually necessary to increase the amount of blur with increasing distance from the zero-parallax plane [28]. Depth estimation along with the blur effect can be achieved using deep learning approaches, for example, DeepLens [273] which is used in the experiments.



(a) without DoF (b) with DoF

Figure 4.29: The ghosting artifacts are visible at the defocused part of the scene (back object). The preprocessed images with DoF effect mitigate the out-of-focus sharp artifacts.

The proposed DoF algorithm needs a screen-space position of a zero-parallax plane point. When the depth map is estimated, a coordinate in the image is sufficient to identify the focus distance. In the case of DoG focusing, the coordinate can be found by generating the focus map as in light field rendering methods [9] based on pixel color deviations. The average coordinates of the focused pixels can lie outside a concave area, so the closest pixel is found from the focused ones, and its coordinates are selected as the DoF focus point.

Multiplication of saliency maps can be used instead of the focus map to identify the focus point. Alg. 12 describes both cases.

Data: The input set of *images*, used focusing *method*
Result: *DoF* images
if *method* **is** *DoG* **then**
    $map = \text{dogFocusMap}(images)$;
else if *method* **is** *Saliency* **then**
    $map = \text{saliencyIntersection}(images)$;
$map = \text{threshold}(map)$;
$map = \text{medianFilter}(map)$;
$avgCoord = \text{averageWhitePx}(map)$;
$coord = \text{closestWhitePx}(map, avgPoint)$;
$DoF = \text{simulateDoF}(images, focusPoint)$;

Algorithm 12: One of the focused pixels is chosen to identify the focus point for the depth of field simulation.

4.3.4 Experiments & Results

15 people participated in the experiments in a laboratory equipped with a 15.6″ Looking Glass 4K model. Users were expected to be LKG users interested in IT in the age range between 20-65. None of the users was diagnosed with serious sight-related problems. All users passed a quick entrance test to exclude people with limited depth perception. They were instructed to read the text on the display and distinguish 2D and 3D parts of the scene. The users were sitting in front of the LKG at an initial distance of 55 cm, as recommended [219]. The average time of one session was approximately 14 minutes with 11 seconds per test. The tests were randomly shuffled to avoid bias when evaluating the same scene or type.

The rendering camera setup is based on Eq. 4.13 from the official LKG documentation[1], where the *travel* value is the length of the capturing camera trajectory, the *distance* is measured from the camera to the geometry of the expected area of interest, and fov is the field of view of the camera.

$$travel = distance \cdot \tan(fov/2) \cdot 2 \tag{4.13}$$

[1] docs.lookingglassfactory.com

Focus Range Identification

The depth range (the distance between the closest and farthest points from the camera) affects the way the user might select the optimal focus distance. If the scene does not contain large depth differences, the zero-parallax plane might be set by the user between multiple objects to keep them all more or less focused. Otherwise, the user would focus on a specific area of interest, leaving the rest of the scene unfocused.

For a given pixel representing a point in a perpendicular distance from the camera plane d, distance from the camera to the zero-parallax plane f, horizontal field of view fov_x, horizontal distance between the two neighboring capturing cameras Δc_x, and horizontal resolution of the image res_x, the displacement of the pixel representing the width of ghosting artifact $ghost$, can be theoretically computed by Eq. 4.14.

$$ghost = \frac{\Delta c_x \cdot (1/f - 1/d)}{2 \cdot \tan(fov_x/2)} \cdot res_x \tag{4.14}$$

In a simple case, the optimal 2D vector representing resolution of one **view** would be calculated from the resolution of the device **screen** and the number of *views* according to Eq. 4.15.

$$\mathbf{view} = \frac{\mathbf{screen}}{\sqrt{views}} \tag{4.15}$$

For a 4K (3840×2140) display with 45 views in a quilt, this would lead to a resolution of 572×322. According to the observations conducted, this resolution is still not high enough, and the improvement in visual quality is visible when the resolution is further increased. The low-resolution artifacts cease to be noticeable at about 800×583. This leads to an approximation of Eq. 4.16, for the optimal **view** resolution calculation from the **screen** one. The vector elements are accessed by the x, y subscripts.

$$\mathbf{view} = (0.21 \cdot \mathbf{screen}_x, 0.27 \cdot \mathbf{screen}_y) \tag{4.16}$$

Note that the ratio between horizontal and vertical resolutions is less than the screen aspect ratio 16:9 and the quilt rows and columns ratio 9:5. This might be caused by the rotation of the lenticular strips. Fig. 4.30 shows the theoretical amount of ghosting for the *Crash* scene where the field of view of the camera is 39.6°, distance between the cameras is 0.050702 meters, and the horizontal resolution is 800 px. The amount of artifacts is increasing not only with a higher distance of the pixel from the zero-parallax plane but also with the zero-parallax plane position being closer to the camera. Scenes focused on very close objects to the camera can be more difficult to setup.

The goal of this experiment is to find the distance between two objects, where users switch from focusing on one object to moving the zero-parallax plane in between them. The testing scene contains two same objects next to each other. The one in the front is static, and the second one starts at the same Z position (depth) as the first one. Then it is moved further away in each measurement. The moving object is scaled in each step so that it covers the same area of the image as the static one (approximately 7.5% of the image). Rescaling is conducted to avoid a perceptional bias that might make users focus more on the larger object.

The measured focus distances were clustered in two groups according to their distance to the optimal focus distance of the closer and further object. The variances of these two groups were averaged and compared to the global variance without clustering in Fig. 4.31.

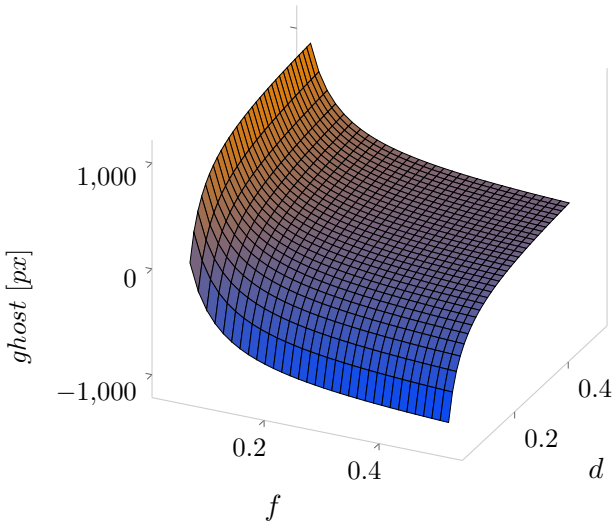


Figure 4.30: The chart shows the amount of ghosting artifacts produced according to the position of the zero-parallax plane f and the depth of the pixel d.

According to the theoretical calculation above, given the zero-parallax plane positioned at the first object and the views rendered in FullHD, with the scene dimensions shown in Fig. 4.32, the artifacts should become visible ($ghost \geq 1$ px according to the plenoptic sampling theory [49]) at the distance from the camera plane of 2.823402 meters. The distance between the object should be $3.01 - 2.823402 = 0.186598$ meters. The graphs start to noticeably diverge around 0.8 meters. The equation is more strict than the real tolerance of the users, but it can be used as a good estimation.

Some users noticed that the object that is closer to the camera is more artifact-prone but also has more 3D-looking when focused. This confirms the assumption shown in Fig. 4.30. Fig. 4.33 reveals that users usually choose the object that looks *more 3D* despite the higher amount of artifacts.

Depth Warping Evaluation

The goal of this experiment is to determine whether the scene can be warped to reduce the depth range. The zero-parallax plane would be closer to more geometry, which leads to more focused scenes. The disadvantage is that the 3D effect and the amount of parallax would be less significant, similarly to cardboard effect caused by the low separation of the capturing camera [291]. The users were choosing a more visually appealing result from pairs of scenes, where they compared differently warped scenes. Fig. 4.34 shows an example of the difference between the warped and original scene on LKG. Given the camera position **cam**, expected $focus$ distance, and the $factor$ of the depth (z axis) shrinking, the warping can be performed for each vertex **vert** of the scene using Eq. 4.17, 4.18, 4.19, producing vertices with new positions $\mathbf{vert}^w$. The x, y, z subscripts access the corresponding vector elements. The objects keep the same relative size in the camera view but are shrinked towards the focus plane position and the camera.

$$\mathbf{vert}_z^w = factor \cdot (\mathbf{vert}_z - focus) + focus \tag{4.17}$$

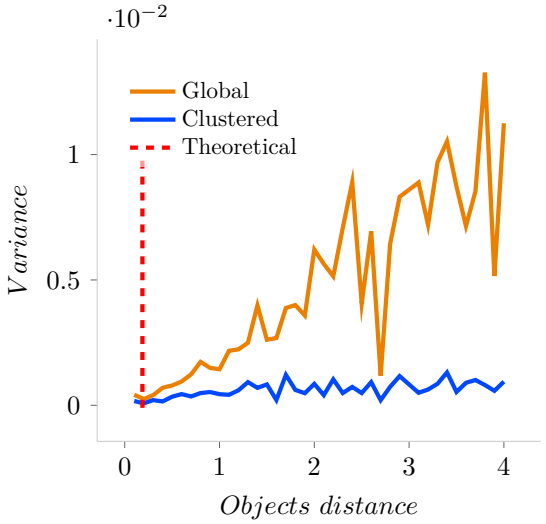


Figure 4.31: The figure shows how the variance of the focusing measurements increases with larger distance between objects. The average variance of the clustered measurements is close to the global variance until a slightly larger distance but close to the theoretical estimation.

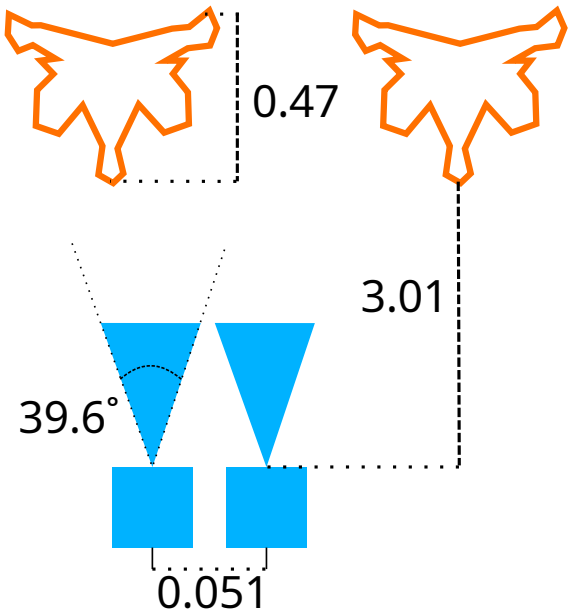


Figure 4.32: Dimensions of the testing *Crash* scene in meters and degrees.

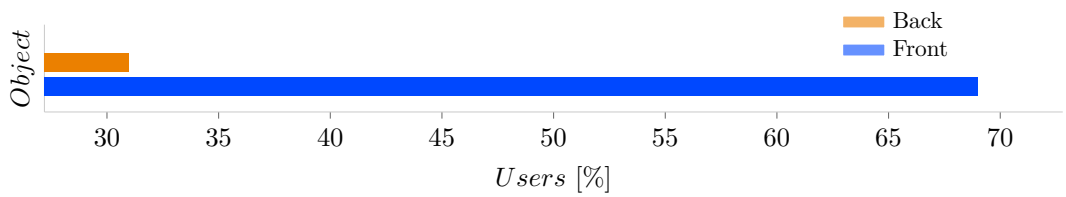


Figure 4.33: The figure shows that most users chose to focus on the front object in the *Crash* scene.

$$\mathbf{vert}_x^w = \frac{\mathbf{vert}_x - \mathbf{cam}_x}{\mathbf{vert}_z - \mathbf{cam}_z} \cdot (\mathbf{vert}_z^w - \mathbf{cam}_z) - \mathbf{cam}_x \tag{4.18}$$

$$\mathbf{vert}_y^w = \frac{\mathbf{vert}_y - \mathbf{cam}_y}{\mathbf{vert}_z - \mathbf{cam}_z} \cdot (\mathbf{vert}_z^w - \mathbf{cam}_z) - \mathbf{cam}_y \tag{4.19}$$

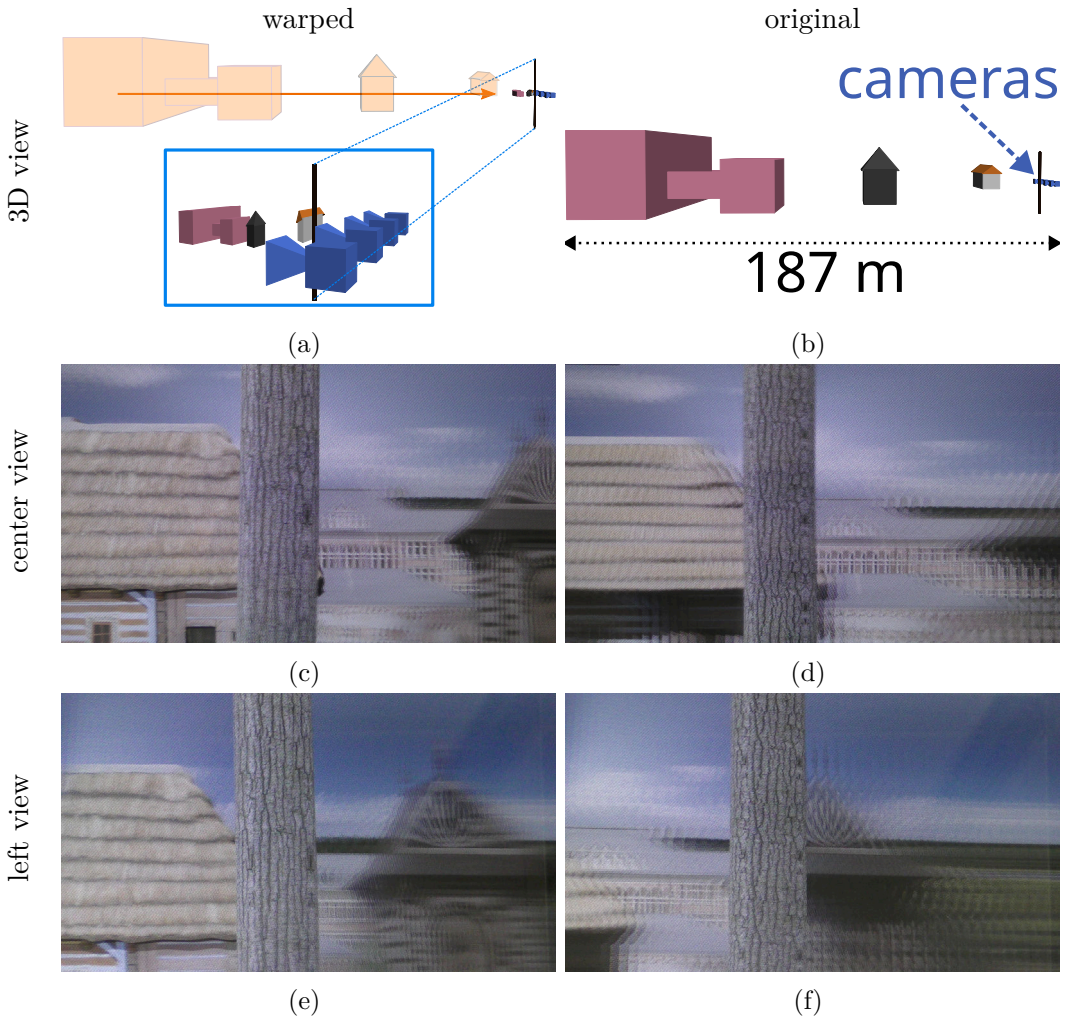


Figure 4.34: The zero-parallax plane is positioned at the tree in the front in all cases. The warped scene has 20× smaller depth range and has more objects behind the tree in focus than the original. However, the parallax is less visible in the warped case.

Fig. 4.35 shows the users' choices when presented in each test with two scenes with different amounts of warping. The users chose the more warped scene almost every time. The only exception was in the case where the original scene was compared to the first warping iteration. The reason is a high amount of artifacts that made both scenes relatively noisy and hard to distinguish which is better. The testing scene was intentionally designed with a large depth range as an expected candidate for the warping method. The measurements prove that the warping of the scene is a suitable approach to improve the quality of the displayed result.

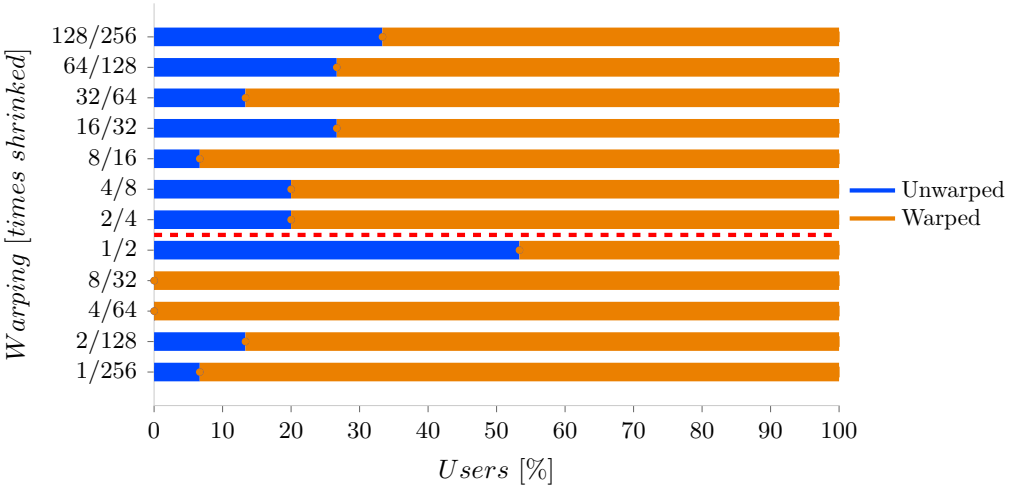


Figure 4.35: The charts show how users prefer higher amount of warping of the scene geometry. The warping reduces parallax but mitigates the out-of-focus artifacts.

Depth of Field Evaluation

The scenes are focused with the saliency method, and then the images are post-processed with a shallow depth-of-field framework [273]. The DoF-based ghosting mitigation assumption is based on the sampling theory, effectively removing frequencies above the Niquist limit [302]. The filtering was applied on the full-resolution input views to avoid aliasing [49]. Users were presented with prefocused scenes in two variants for each scene, with and without depth of field. Their task was to choose which of each pair is more appealing in terms of visual quality and 3D experience on the LKG. The prefiltered scenes were manually validated.

Surprisingly, Fig. 4.36 shows that most users chose the scene without DoF as a more appealing result. Therefore, DoF cannot be used without a prior analysis of the content of the scene. Although the interperspective aliasing artifacts are smoothed, the added blur can decrease the visual comfort of the imagery on LKG. Fig. 4.37 shows the results for each scene. The *Cars* scene was mostly preferred with DoF, the *Knight* scene without, and the expected optimal candidate for DoF, *Angel*, was slightly more preferred without DoF; see Fig. 4.38. Few observations can be derived from the measurements:

- DoF reduces the quality of the result in scenes with small depth range, for example, *Knight, Table*.
- The best choice for DoF is a scene with sharp edges that are only slightly out-of-focus, for example, *Cars, Room*.
- Applying DoF on a distant background does not always improve the result, for example, *Angel*.

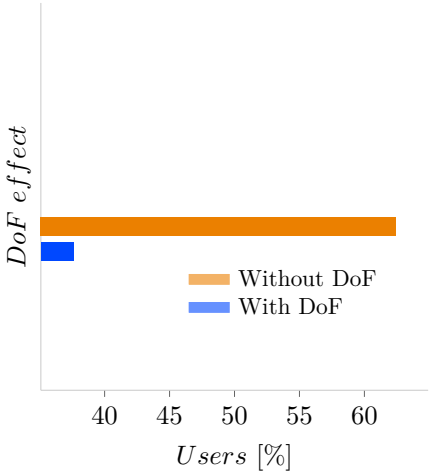


Figure 4.36: The overall amount of users that preferred original or DoF scene across all tests and scenes.

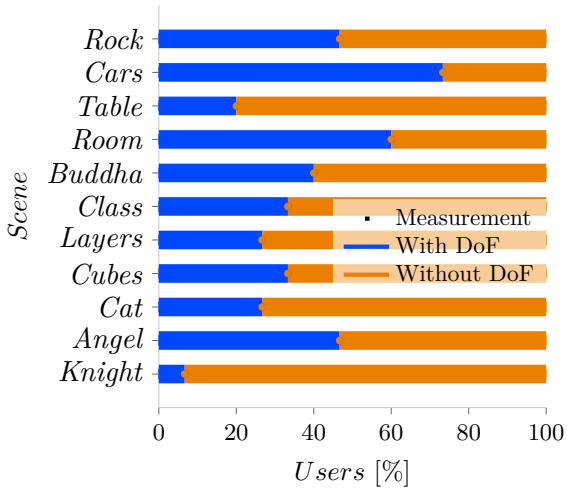


Figure 4.37: The chart shows the user preferences regarding the DoF effect for all tested scenes.

Focusing Method Comparison

The edge detection (DoG) approach selects as the optimal zero-parallax plane position a distance where most of the scene is focused. The method does not take into account human perception. The saliency method focuses on a specific area in the scene that is most likely to be looked at by the user. The difference is shown in Fig. 4.39.

The autofocusing methods are time-consuming due to the scanning process, where all input images must be analyzed, filtered, and evaluated for all steps of the focus range. A measurement was carried out to analyze the performance of the methods in relation to the resolution of the input images. Fig. 4.40 shows the execution times of both methods and the resulting focus distances. The measurement was carried out on an Arch Linux machine equipped with Intel® Core™ i5-8500 CPU 3.00 GHz and 16 GB DDR4 RAM.

The density of the sampling range affects the performance/quality ratio linearly because it only adds more iterations. The density is set to 400 steps in the $[-1; 1]$ range, which was empirically identified as sufficient to cover the entire dataset.

The DoG method is approximately 6× faster than the saliency one, but the detection of a focus point for the DoF simulation is approximately 30× faster in the saliency method. However, the DoG method is faster in the total time comparison, because the detection of the point takes only 15% of the total DoG time and 0.09% of the saliency time. The saliency method is stable in all resolutions, whereas the DoG method slightly changes the result in different resolutions and returns a completely wrong focus distance for resolutions below 512×288.

The saliency method had problems in four scenes (*Brick, Cubes, Rock, Cars*). The reason is that the generated saliency maps did not mark the same objects in the scene as salient. The DoG method always returns the best overall focus distance.

The experimental task for the users was to move the slider to optimally focus the scenes. The user might move the zero-parallax plane to an object of interest or to the position that makes a bigger part of the scene focused, sacrificing the sharpness of other parts.

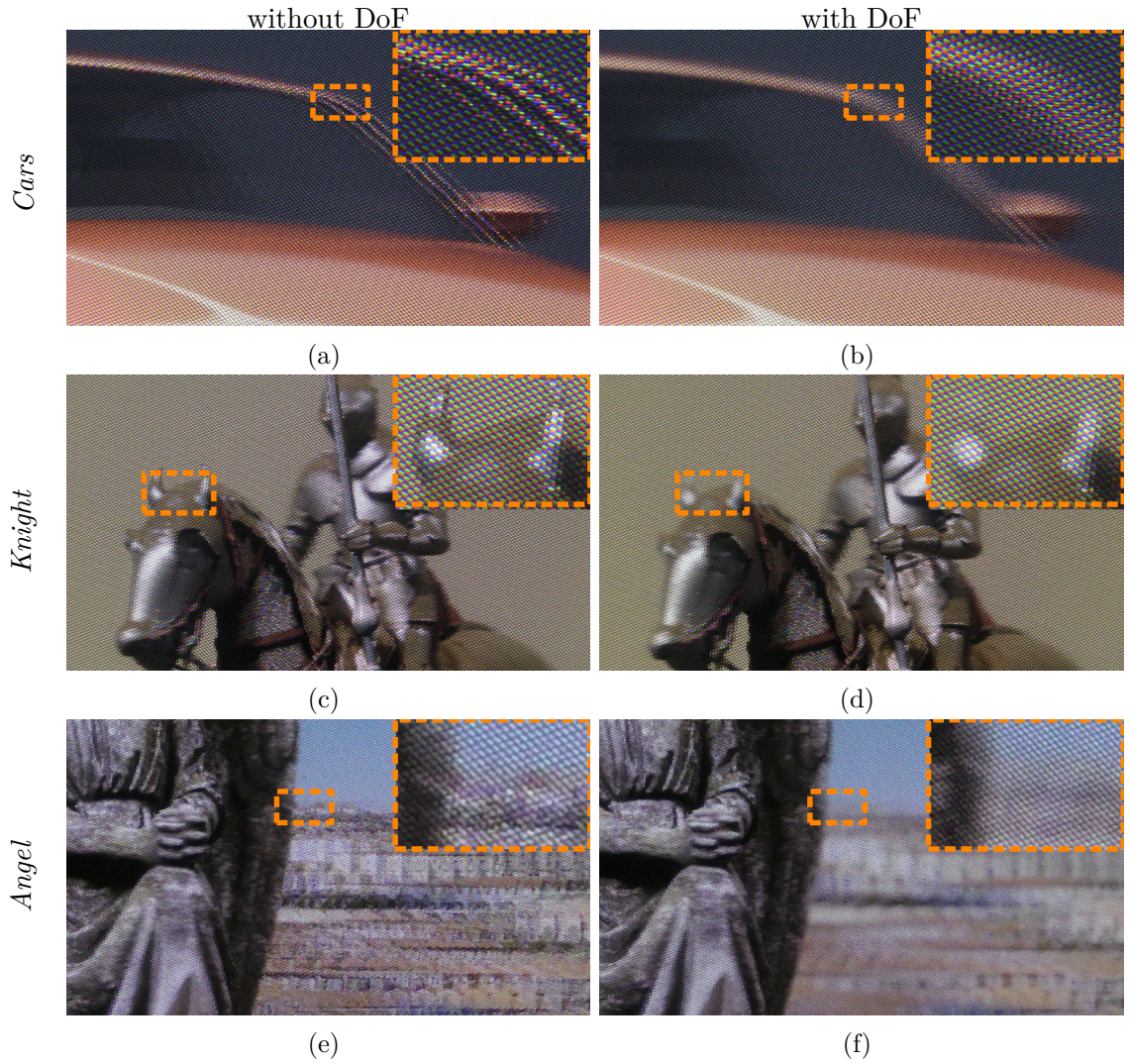


Figure 4.38: Examples of the most interesting scenes details on LKG with and without DoF.

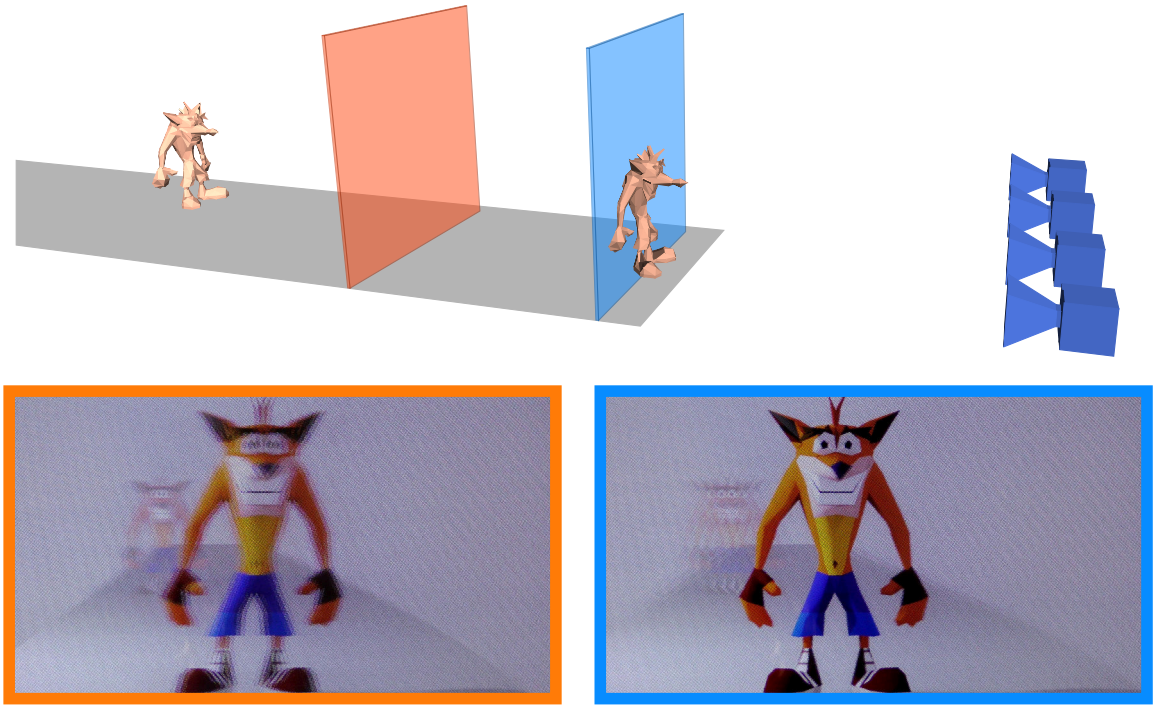


Figure 4.39: Two zero-parallax planes are depicted in the 3D scene, one is aligned with the front object which is regarded the object of interest as the result of the saliency method and the second one is roughly between the objects (a bit closer to the one with larger area in the final image) trying to find the best average focus distance as the result of DoG method.

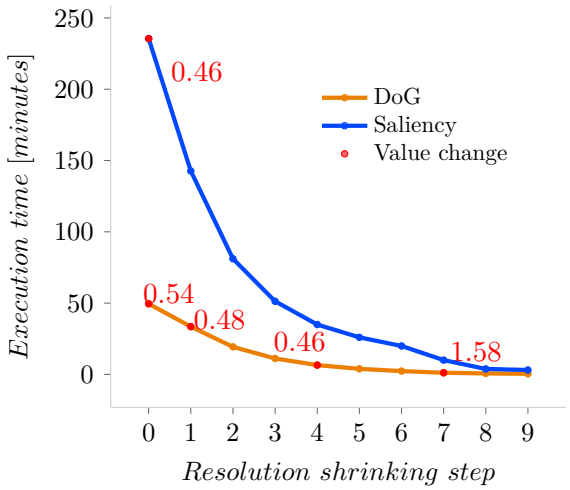


Figure 4.40: The figure shows the performance of both autofocusing methods. The resolution of the input images shrinks by 3/4 of the previous one along the x axis, starting at 4K. The range was scanned with 200 steps on the *Crash* scene. The red nodes indicate how the value of the focus changed.

Fig. 4.41 shows that the saliency method outperforms the DoG one and simulates the human choice more precisely with respect to the overall error. The total number of measurements that were closer to the DoG method is higher; see Fig. 4.42.

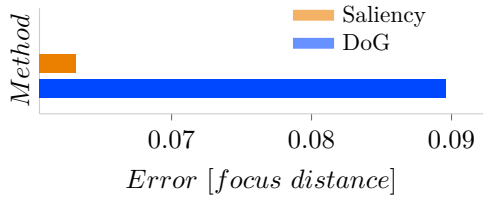


Figure 4.41: The chart shows the average difference of the automatically detected focus distances from the users' measurements.

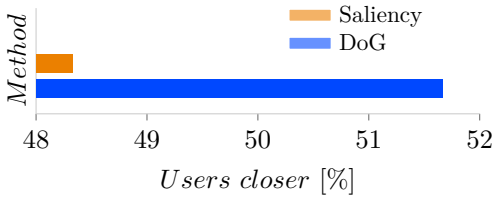


Figure 4.42: The bars show how many of the users measured the focusing closer to the given method in 180 measurements.

Note that in some cases, such as in many measurements of *Knight* scene, the measurements of both methods were very close. Therefore, the error is taken as the most representative metric. Fig. 4.43 shows the average, minimal, and maximal measured values and values which were automatically detected by the two proposed methods for each scene. The DoG method is not optimal for scenes with a large depth range and scenes with a lot of high-frequency features. The saliency method performs slightly worse in scenes with low frequencies and large solid-colored areas.

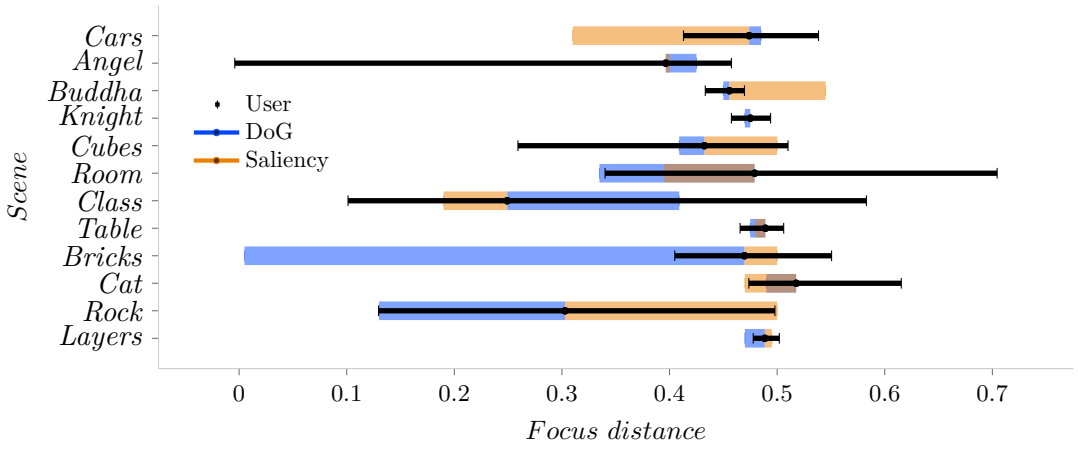


Figure 4.43: The automatically detected focus distances of both methods and the average measured focus distances with their minimal and maximal values are visible in the chart for each scene.

The DoG method serves as a good first guess for the focusing, expecting the user to adjust the result for desired result. The saliency method is a better approach for reaching an artistic result without further adjustments. The experiments show, that a general focusing approach would consist of both methods, first performing the saliency one. If the accuracy of the saliency method is too low, for example, low amount of overlapping saliency masks, the DoG method can be used as a fallback. This approach would improve the accuracy of the focusing, compared to the saliency method, by more than 50% according to the measurements.

4.4 Capturing Camera Trajectory Distortion

The main goal of this section is to discover the limitations and most crucial parameters related to the shape of the capturing camera trajectory when creating input views for 3D display. An evaluation method was proposed and experimentally measured in a user study [6]. This study reveals specific attributes of 3D displays that define the limitations for the input light field data. For example, the proposed light field rendering method can be used accordingly.

The linear trajectory of the cameras was distorted in various ways. The users were asked to adjust the amount of the distortion to the maximum acceptable values while evaluating the visual quality of the result on the LKG; see Fig. 4.44. Some distortions of the trajectory might affect the resulting visual quality in a different way or with a different impact on the human visual system than others.

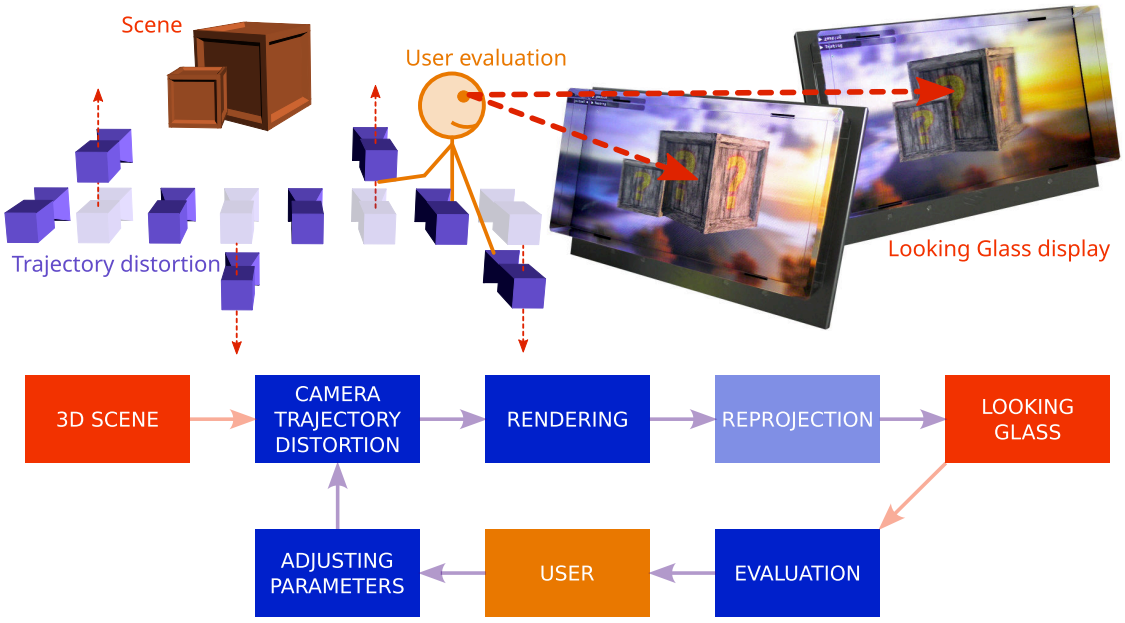


Figure 4.44: The testing 3D scene was captured by cameras placed along the optimal trajectory. The views obtained were displayed on a Looking Glass Factory 3D display. The users adjusted the amount of simulated distortions of the trajectory. Each distortion type was examined in a separate measurement. The users evaluated the resulting 3D experience on the display and were instructed to set the amount of the distortion to the maximal value where the possible visual discomfort was not noticeable yet.

Users might capture the scene with a handheld camera or extract the scene from arbitrary video. The discovered limits might be used to produce a score of scene suitability and estimate the level of visual comfort for such a scene. The perfect camera trajectory is difficult to achieve in real life without special equipment. Proprietary software, such as games, can use tools like NVIDIA Ansel to enable restricted free-look camera view. The proposed guidelines might help to correctly setup the capturing process when intentionally creating suitable views and also when processing already existing videos or photos which are not suitable to display on LKG without additional edits. A 2D-reprojection-based correction method of the images is also proposed and evaluated. The method is capable of improving

the visual quality of the unsuitably captured scenes, thus loosening the required capturing rules. The limits discovered might lead to a more comfortable experience in the field of 3D displays, which had been gaining popularity and various applications in recent years [292].

Compared to previous studies, more camera trajectory distortions were tested with additional adjustments of user behavior and scene content. More distortion types (translation and rotation in 3D and their combination) were implemented in this study, and they were also measured in more variations (with or without background, with user's head moving and being static). The study does not focus on possible misperceptions and depth accuracy issues [108] or visual discomfort caused by specific disparity-related problems [258]. The measurements discovered limitations valid for the current state-of-the-art 3D display that might differ from previous tests with older devices. The device in this study is not experimental as devices in previously published papers, but it is a widely used consumer 3D display device. Other technologies, such as Lumepad 3D tablet, provide only a limited functionality of LKG and the results would be similar, so only LKG was considered during the testing.

4.4.1 Evaluation Method

All experiments require the users to adjust the normalized slider in the $[0, 1]$ range, which affects the distortion of the capture camera. At the same time, they visually evaluate the result on the LKG. In this way, users find the optimal or maximum value of the parameter, keeping the result visually acceptable. The four main objectives of the experiments are as follows:

1. Find the maximum acceptable amount of the distortions.
2. Identify the most crucial and the least crucial type of distortion.
3. Reveal the effect of the content of the scene or user behavior on the visual quality of the distorted scene.
4. Measure the efficiency of 2D reprojection of the images that compensates for the distortions.

An OpenGL application with a simple 3D scene with two boxes and a background plane was implemented to conduct the experiments; see Fig. 4.45. The zero-parallax plane distance was set to keep most of the area containing the boxes in the middle of the scene focused. Fig. 4.46 is the blueprint of the scene. The experiments were conducted with 10 people in a laboratory equipped with a $15.6''$ Looking Glass 4K model, rendering 45 views of the scene. Users were expected LKG users, people interested in IT, in the 20-40 age range with no serious sight indisposition. During the experiments, the participants were sitting in front of the LKG at an initial distance of 55 cm, which is within the recommended viewing distance range [219]. A supervisor noted the relevant comments of the users. The results were statistically processed and averaged.

During the first set of measurements, users were instructed not to move their heads to evaluate only the artifacts caused by the interpolation of the views. The use case for this scenario is, for example, watching a 3D movie where users are expected to sit still. During the second set, they moved their heads from side to side to see the parallax and to evaluate the quality of the seeming motion of the objects in the scene. The user might do so, for example, when playing games or exploring 3D models.

Figure 4.45: 2D view of the testing application where the user's task was to move the slider to the maximal value while evaluating the result on the LKG.

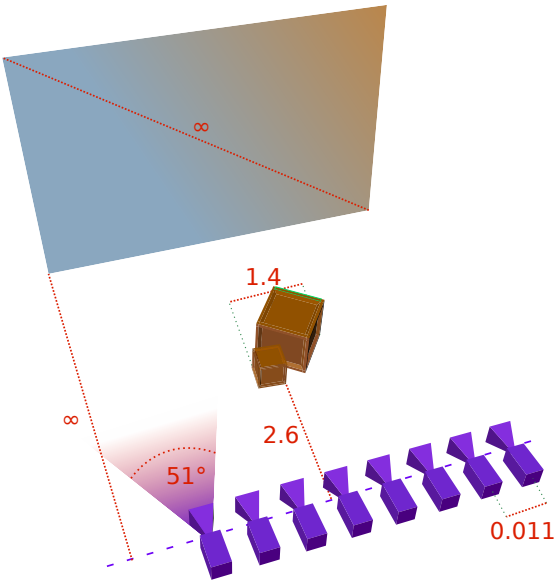


Figure 4.46: The figure depicts the 3D scene used in the experiments. The measures are in the world space units. The cameras are set according to the official LKG recommendations.

The user might focus solely on the main object in the scene (boxes) or may also evaluate the artifacts in the background. Therefore, another categorization was used, using the same scene with and without the background. The categories are depicted in Fig. 4.47. A total of 128 measurements were conducted with two output values for each measurement, the slider value, and the time duration of the particular measurement. At the end of the session, users were asked additional questions about the experiments:

1. What distortions were the most problematic and why?
2. What kinds of artifacts did the participant notice?

Trajectory Distortion

The shape of the capturing camera trajectory can be described by a function. Different shapes might result in different kinds of artifacts and user experience, even if the underlying camera transformation is the same. Theoretically, n basis functions f_i can be defined and combined with weights w_i to simulate a large set of camera trajectory shapes according to Eq. 4.20. The resulting xth camera displacement $\mathrm{d}(x)$ can be calculated by multiplying the amount of transformation $\mathrm{t}(x)$ and the weighted sum of the basis shape functions.

$$\mathrm{d}(x) = \mathrm{t}(x) \cdot \sum_{i=0}^{n} (w_i \cdot \mathrm{f}_i(x)) \tag{4.20}$$

Two basis functions were defined and used separately (only one weight is 1 and the second is zero): a sine function to simulate camera shaking and a linear function to simulate gradual camera change. Therefore, the distortions were implemented in two variants:

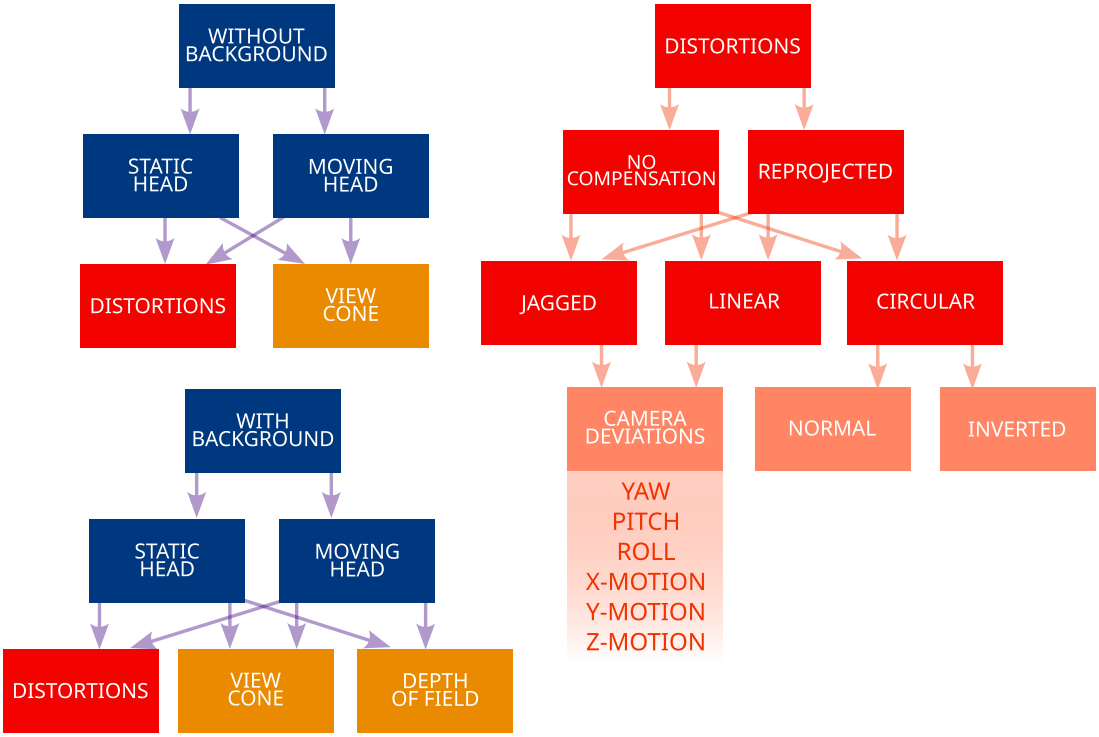


Figure 4.47: The scheme describes the categories of the measured tasks in the experiments.

- **Linear** where each camera is affected by the distortion relative to the previous one (incrementally).
- **Jagged** where the cameras are distorted in an alternating pattern.

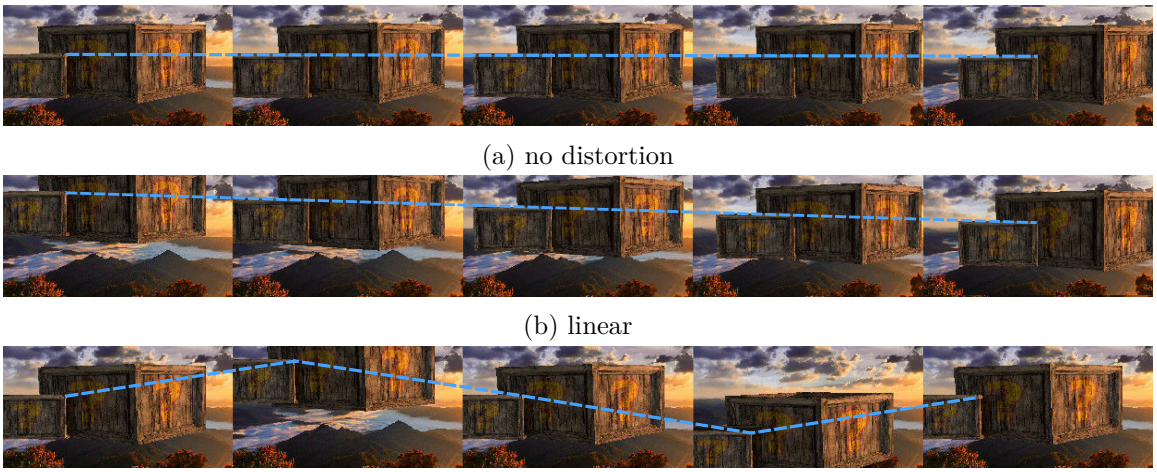
Fig. 4.48 shows how the cameras are placed. Fig. 4.49 is an example of the resulting quilts. Six basic camera transformations were implemented: rotation and translation on the three basic axes; see Fig. 4.50.

(a) no distortion (b) linear (c) jagged

Figure 4.48: The distortions can be applied in two modes, linearly with an offset relative to the distance from the first camera and jagged where the camera moves up and down repeatedly. Vertical translation is used here for the demonstration.

Linear distortion causes more ghosting in the static view; see Fig. 4.51. The motion artifacts, when moving the head around the display, are continuous. The objects move in the direction of the distortion smoothly; see Fig. 4.52. The jagged distortion creates less ghosting. Two images of three might be placed on the horizontal line in the best case. However, the motion of the objects suffers from wobbling artifacts; see Fig. 4.53. All proposed tests simulate the distortions that the user might create, for example, when capturing the scene with a handheld camera.

An extra distortion is a circular shape of the trajectory. Two versions are used, a circle around the objects where each camera points to the center of the circle and an inverted



(a) no distortion

(b) linear

(c) jagged

Figure 4.49: The figure shows different kinds of distortions. Each line shows a subset of views from the whole quilt. The blue line tracks the position of the same 3D point.

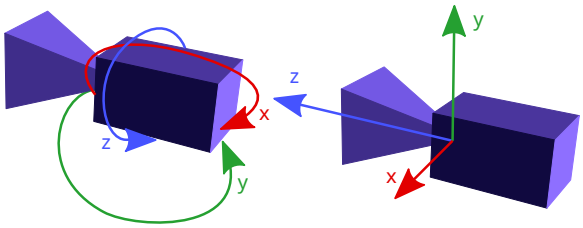


Figure 4.50: The basic transformations used to distort the trajectory: translation and rotation (roll, pitch, yaw) in X, Y, Z axes.

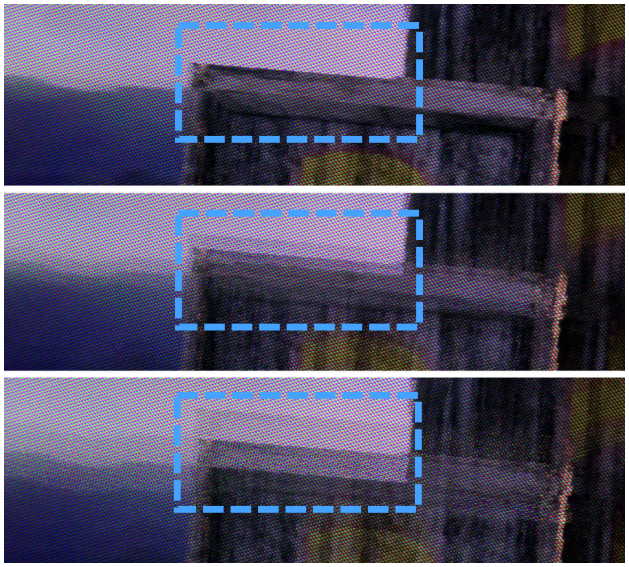


Figure 4.51: The figure shows the LKG imagery captured with the ideal camera trajectory in the top row and increasing jagged vertical distortion below. The ghosting effect is visible at the edges of the objects.

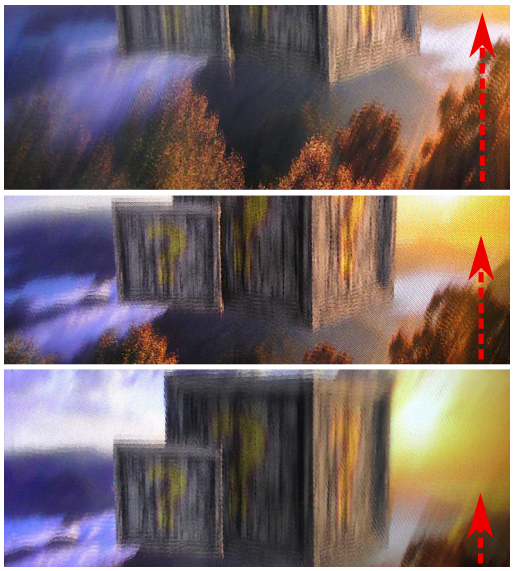


Figure 4.52: The figure shows the LKG view from rightmost, center, and leftmost viewing angle. The objects move down according to the linear distortion with a motion blur.

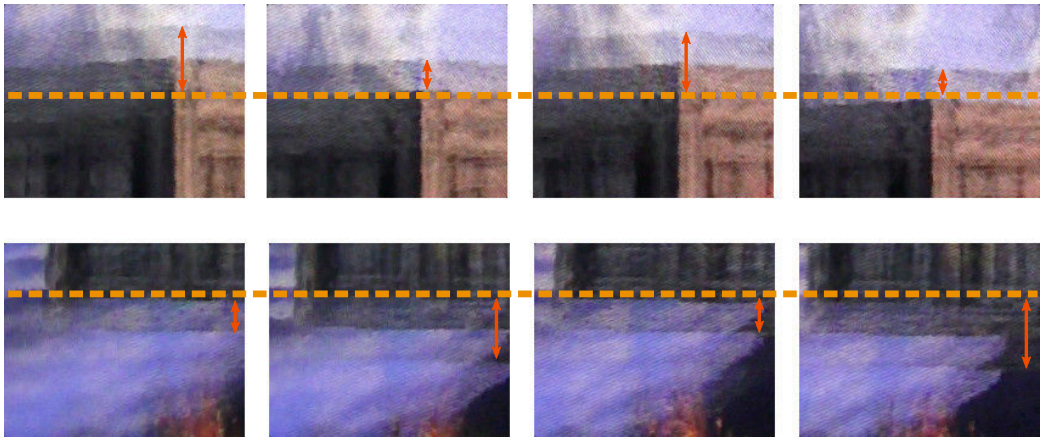
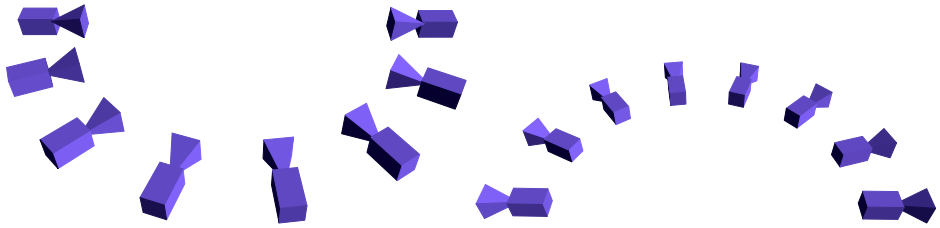


Figure 4.53: The figure shows two zoomed areas of the LKG imagery captured from different but close angles. The jagged vertical distortion leads to wobbling artifacts of the edges. The lines represent the expected edge. The ghosting artifact oscillates around the expected edge.

one, where the cameras point outside; see Fig. 4.54. This simulates the case where the user might circle around the object with a camera to capture it from all angles or when capturing the scene standing at one place and rotating the camera around, such as when taking a panorama photo.



(a) circular (b) inverted circular

Figure 4.54: The circular distortion interpolates the camera positions between the original and the circular trajectories. Two variants are implemented, half-circle around the object where the cameras point at the center and inverted half-circle as if taking a panorama photo.

Extra Measurements

Additional measurements were included. The first was to discover the limits and optimal value of the view cone (distance between the cameras). According to the documentation, 35° is the optimal value of the total bounding view cone for all cameras combined according to the physical parameters of the display. This experiment was conducted to verify this claim and to find the possible range of acceptable view-cone values.

Two additional tests, with the scene using a 30° and 60° view cone, measuring the depth-of-field effect were also included. These angles were chosen to represent narrow and wide view cones. Users were asked to set the amount of background blur (Gaussian blur radius) that can mitigate out-of-focus artifacts [133].

The default view angle of the camera from Fig. 4.46 is used in all measurements except for the depth-of-field test, where the angle is changed to test the effect of the wide and narrow view, and in the view cone test, where the users change it directly.

Reprojection

All distortion measurements had corresponding versions with a 2D correction of the images back to the optimal position. The pixels of the rendered 2D images are completely reprojected by the inverse transformation of the distortion with a specified constant depth. The goal of these tests is to evaluate the efficiency of this correction process in fixing inappropriately captured scenes. This is the evaluation of the ideal case, where the transformations are known: the view matrix $\mathbf{view}_{opt}$ of the camera in an optimal position, the distorted view matrix $\mathbf{view}_{dist}$, and the projection matrix $\mathbf{proj}$. The reprojected coordinates $\mathbf{cr}$ are obtained in Eq. 4.21 where $\mathbf{cv}$ are the view-space coordinates of the pixels in the distorted image. The inverse matrix of the distortion is used to correct the image and transform it according to the optimal position of the camera.

$$\mathbf{cr} = \mathbf{proj} \cdot \mathbf{m}_{dist}^{-1} \cdot \mathbf{cv} \tag{4.21}$$

The distortion is represented by a transformation matrix $\mathbf{m}_{dist}$, obtained as described in Eq. 4.22, as the difference matrix of the optimal view matrix and the distorted one.

$$\mathbf{m}_{dist} = \mathbf{view}_{dist} \cdot \mathbf{view}_{opt}^{-1} \tag{4.22}$$

The $\mathbf{cv}$ denotes the view space coordinates acquired by Eq. 4.23 from the input normalized texture coordinates (NTC) $\mathbf{co}$ of the image. $\mathbf{1}$ denotes vector filled with the value one.

$$\mathbf{cv} = \mathbf{proj}^{-1} \cdot (2 \cdot \mathbf{co} - \mathbf{1}, d, 1) \tag{4.23}$$

The coordinates must first be converted to the normalized device coordinates (NDC). A predefined depth d that corresponds to the view-space depth of the focused objects (boxes) is used as a constant global depth for all pixels in the image. The projection is then reversed using the inverted projection matrix. The coordinates need to be divided by their fourth element to compensate for the perspective-divide operation; see Eq. 4.24. The subscript (x, y, z, w) denotes the selection of the given coordinates.

$$\mathbf{cv} = \frac{\mathbf{cv}_{xyzw}}{\mathbf{cv}_{w}} \tag{4.24}$$

Eq. 4.25 is used to perform the perspective division and to get the coordinates back into NTC, resulting in the final $\mathbf{cr}$.

$$\mathbf{cr} = \left(\frac{\mathbf{cr}_{xy}}{\mathbf{cr}_{w}} + 1 \right) \cdot \frac{1}{2} \tag{4.25}$$

In practice, 3D reconstruction methods such as ORB-SLAM3 [47] would be necessary to estimate the camera positions and their distortions from the optimal path. Views in the quilt should also be 2× zoomed in to hide the areas at the edges of the reprojected image, where visual information is not available. Fig. 4.55 shows an example of the reprojection.

(a) The images are reprojected without zoom and crop.

(b) The images are reprojected with zoom and crop.

Figure 4.55: The figure demonstrates the effect of the reprojection of the 2D images into the correct positions. The orange line marks the area with missing image data. The images are zoomed to mitigate border artifacts where the image was not rendered.

4.4.2 Experiments & Results

The user experience measurement criteria [265] do not focus on the general usability or efficiency of the product in these experiments. The time of each task or the overall opinion of the user about the LKG is not important. The main metric is the visual appeal of the result and the visual comfort related to human perception, based on the amount of distortion. The effort of the user to complete the task was also checked and noted to identify possibly problematic types of distortion.

The duration of the session for one participant was approximately 29 minutes, where one measurement took 14 seconds on average. Half of the participants felt slight fatigue after the experiments, but none reported sickness, such as when watching uncomfortable virtual reality imagery. The most significant artifacts according to the users were vanishing edges, motion-blur-like noise, and ghosting. When evaluating the results while moving their heads, unrealistic motions of the objects and woobling artifacts were reported. Tab. 4.3 explains the names of the measurements used in this section.

name	description
linear	The cameras are distorted incrementally. nth camera is distorted n times.
jagged	The cameras are distorted alternately. Positive direction, no distortion, negative direction, no distortion, etc.).
static	The participants were instructed not to move their heads to evaluate only the stereo image.
moving	The participants were instructed to evaluate the result from different viewing angles.

Table 4.3: The table explains the terms used in the measurements.

The units used in the measurements are listed in Tab. 4.4 that shows how much the view image changes when its camera is transformed. Two metrics are used: SSIM over the whole image compared to the undistorted one and displacement of one pixel placed in the area of interest, on the zero parallax plane. The pixel is placed at the top left corner of the smaller box. This pixel was chosen because of its position at the edge where most of the artifacts are noticeable. Note that other pixels in the scene might move with a different velocity due

to the perspective projection, their depth, and the nature of the applied transformation. SSIM metric was chosen as a commonly used standard image similarity metric which better reflects the image differences for the human observer than other metric such as PSNR [58].

transformation	amount	axis	SSIM	displacement [px]
rotation	1°	x	0.636040	19
		y	0.596173	19
		z	0.651903	3.2
translation	0.01 ws unit	x	0.850915	5
		y	0.891480	5
		z	0.938095	2

Table 4.4: The table shows the relationship between the given transformation and the difference in pixels between two neighboring 1920×1080 views. The pixel displacement is the length of the trajectory that the chosen pixel (a corner pixel in the area of interest) traveled due to the applied transformation. Identical images would be evaluated by SSIM value of 1. The world-space units in the 3D scene are labeled as ws units.

The main questions defined in Section 4.4.1 are answered as follows:

1. The maximum distortion should generally not be greater than 0.3° in rotations and 0.02 world units (4 px in the area of interest) in translations.
2. Users mostly tolerated rotational distortions followed by translational and circular. The jagged variant is also more artifact-prone than the linear one.
3. A featureless or solid background can, in some cases, increase the tolerance to distortions. The more users move their heads around the LKG, the more unpleasant the distortion artifacts appear.
4. The 2D reprojection can be used to significantly improve the quality of the distorted quilts, up to 60 times in the best cases.

The pixel difference between the distorted and optimal quilt is not enough to predict the effect of the distortion on the resulting visual quality. Fig. 4.56 shows the SSIM metric values between the optimal and distorted quilts according to the measured limits.

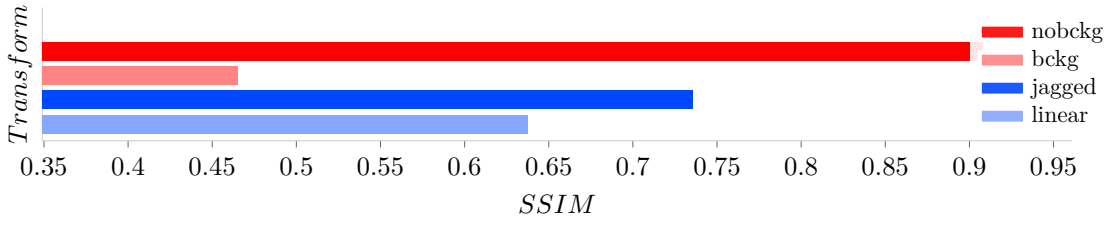
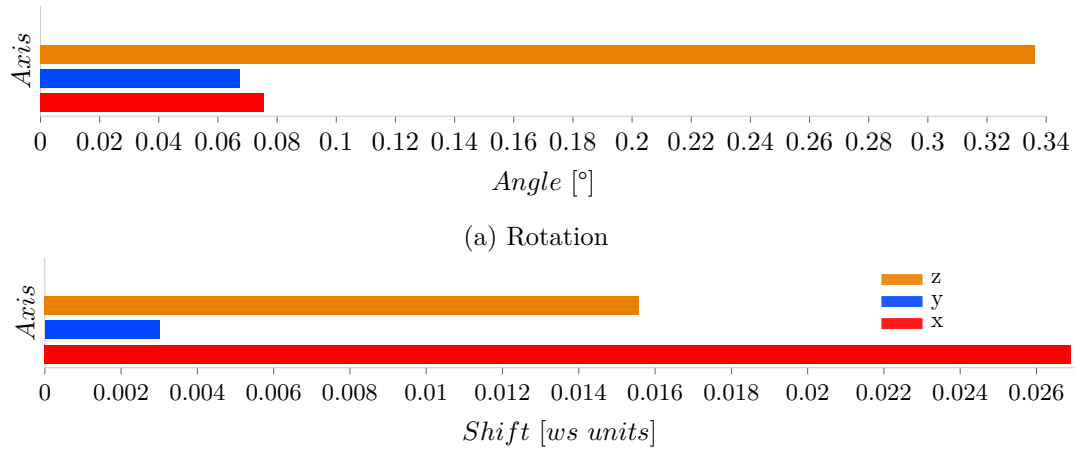


Figure 4.56: The chart shows SSIM comparison between the optimal and average distorted quilt. A lower value means higher tolerance.

In this way, the measured results are normalized by the image difference and different transformations can be compared. Users tolerate a higher amount of rotational than translational, and linear over jagged variants of the distortions. The presence of background

does not show a significant difference in this overview. The z axis rotation is the most tolerated, while the rotation of the x and y axes shows very similar limits. The same facts are not valid for translations, where the x axis has the highest distortion limit, followed by z, and leaving y as the most artifact-prone transformation; see Fig. 4.57. Visually, a small amount of y rotation is similar to x translation and x rotation to y translation. Despite that, the results do not correlate due to the parallax effect present in the translation.



(a) Rotation

(b) Translation

Figure 4.57: The charts show the average measured limits of the translations and rotations along the three basic Cartesian axes. Translation along the x axis and rotation along the z axis are the ones that users tolerate most.

Reprojection can significantly loosen the limits according to Fig. 4.58. This correction of the distorted images is most efficient at rotational distortions because of the absence of parallax-related artifacts compared to the translational ones. Generally, the reprojection can be used in almost all cases to correct the distorted quilt by hundreds of percent.

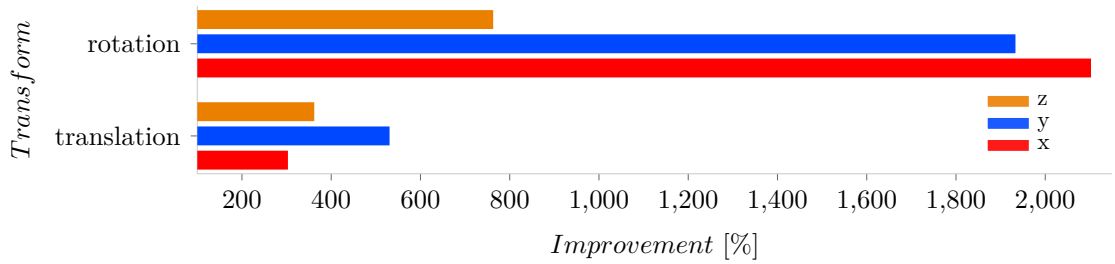


Figure 4.58: The effect of the reprojection is demonstrated in this chart. The percentage improvement of the measured limit is most significant in rotations, especially around the x axis.

SSIM Comparison

The optimal quilt was compared, using the SSIM metric, with the quilt acquired with the amount of the distortion corresponding to the average measured value. This ensures an objective measurement of the difference in the views after the trajectory distortion; see Fig. 4.59. The chart indicates two main facts. Users tolerate more pixel difference in:

- rotations than in translations,
- linear distortions than in jagged ones.

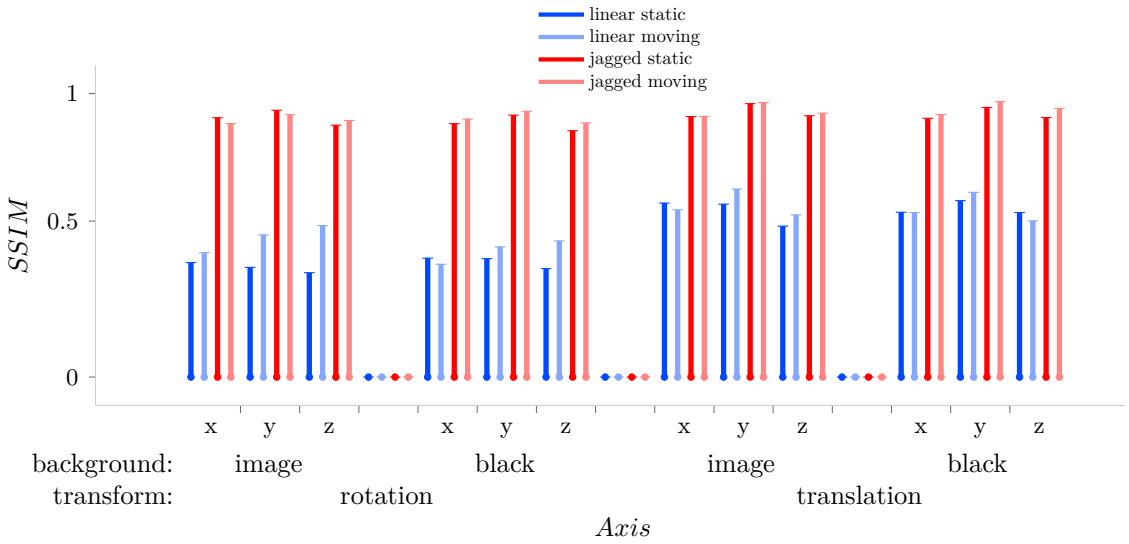


Figure 4.59: The chart contains SSIM values of the distorted quilt compared to the optimal. The amount of the distortions corresponds to the average values from the measurements. The lower the value, the greater the tolerance of the participants for the given camera distortion.

This result proves that the perception of the image on the LKG does not depend only on the pixel difference, but also on the type of transformation which might affect how the human brain interprets the artifacts. Note that the values in the chart would not differ if the pixel difference of the quilt was the only parameter that impacted quality. The high SSIM values of jagged variants of the distortions are also caused by the fact that some images are the same as in the optimal quilt. It might be surprising that the participants did not tolerate even higher distortions in the rest of the images in the jagged variants. Apparently, the resulting quality depends on each of the views and cannot be evaluated globally over the whole quilt because the imagery the user perceives is always a mix of three or more views. In most of the cases, the motion of the users' head also lowers their tolerance to the artifacts.

Rotations

Fig. 4.60 summarizes the measurements of the maximum angles. The participants marked the z rotation as an unexpected behavior of the objects according to the position of the head. The difference between the static and moving variants is the most significant. Nevertheless, the measurements show that this rotation allows for the most freedom. The values obtained are generally slightly lower than the results of other studies on stereoscopic images [243]. However, certain cases show greater tolerance, for example, the static jagged variant of the z rotation without background image. The linear variant of the tests shows higher user tolerance in the evaluation without head motion. The situation is the same in the jagged tests with a black background.

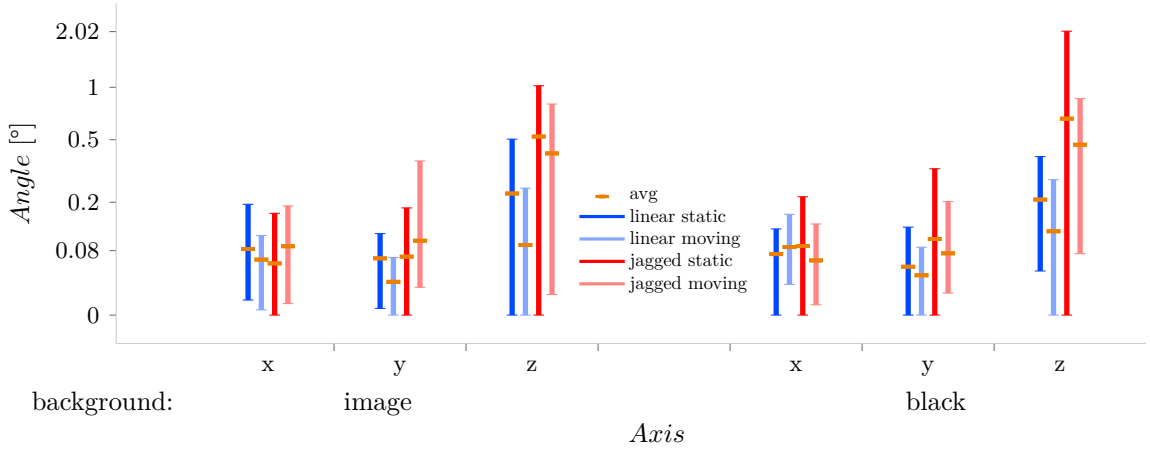


Figure 4.60: The chart shows the maximum rotational angles in all three main Cartesian axes that the users marked as visually acceptable.

Translations

The results in Fig. 4.61 reveal that the translational distortion in the x axis is quite robust to the artifacts, followed by the z axis, which visually creates a zoom effect in the view. The x axis translation moves the view along the direction of the optimal line trajectory, which corresponds to the orientation of the human eyes. The z axis distortion might look more natural to the user than the y axis, because the vertical motion of the objects on the y axis resembles shaking of the 2D imagery, while the zooming of the objects might resemble the deformation caused by a perspective projection. The sitting position of the participants also reduces the expectation of the vertical motion of the rendered objects and does not exclude the possible motion of the user's head closer and farther away from the screen, which resembles the z axis distortion.

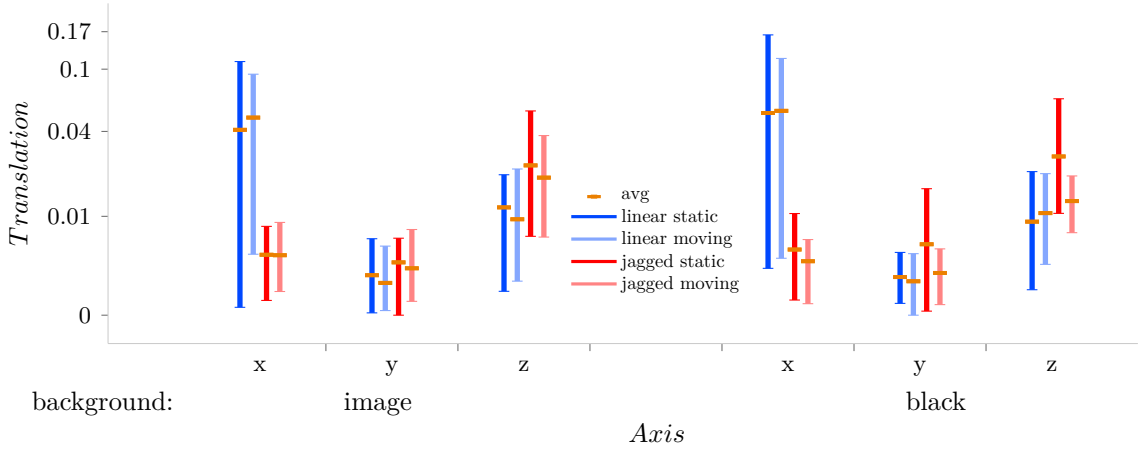


Figure 4.61: The chart shows the maximum amount of translation in the world space units in all three main Cartesian axes that the users marked as visually acceptable.

Reprojection

The 2D reprojection of the view images can significantly increase the maximum limit of the transformations; see Fig. 4.62. The reprojection is most efficient at rotational distortions because of the lack of parallax effect. As expected, the removal of the background significantly increases the efficiency of the reprojection in almost all cases. The expected depth used in the reprojection process (Eq. 4.23) corresponds to the zero-parallax plane that is placed at the boxes. The best cases for reprojection ($45-60\times$ improvement) are the jagged variants of x and y axis rotations. The most significant improvement was detected at the inverted circular trajectory ($30\times$).

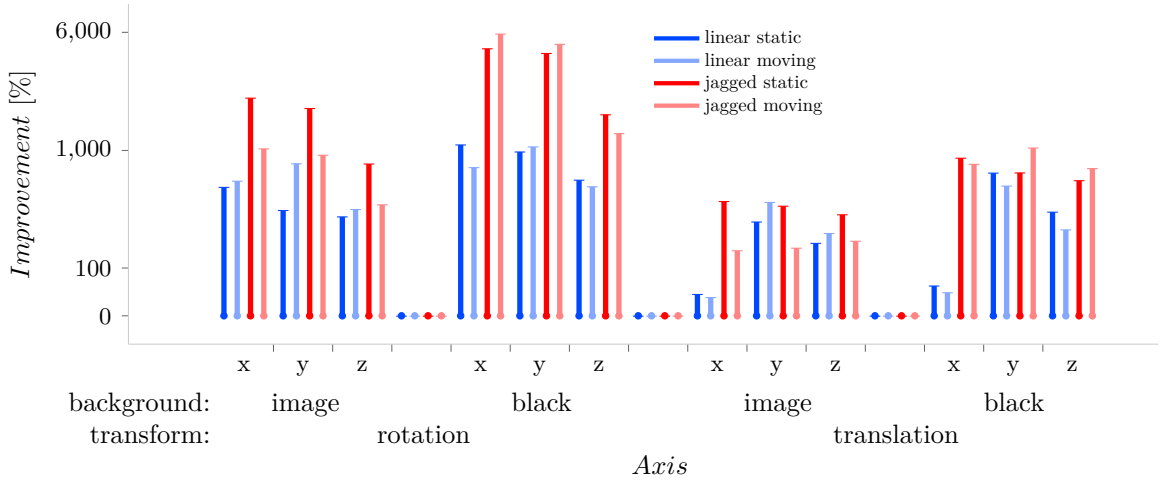


Figure 4.62: The chart shows the average improvement of the measured limits at each transformation after applying the 2D reprojection to correct the distortions. The original limit is considered to be 100%. The values are calculated from the average measured limits in each category.

Extra Measurements

The linear trajectory can be interpolated into half circle only by the factor of 0.08 (1 = half circle), which proves that circular capture is unusable. Users marked the inverted circular distortion as the most artifact-prone transformation of all measurements.

For a simple stereoscopic image, where the user is not expected to move around, the view cone can be lowered almost to half of the recommended value for the background scene. According to the measurements, the view cone value can generally be set in $[13°; 50°]$ range.

The participants mostly set the value of the depth of field low enough to mitigate the ghosting artifacts. The optimal amount of blur seems to be directly proportional to the view-cone value. Doubled view-cone requires doubled radius of the blur. Users tended to increase the blur when head motion was allowed; see Tab. 4.5.

view cone [°]	head	avg radius
30	static	0.562100
	moving	1.865934
60	static	2.287247
	moving	2.383613

Table 4.5: Measured average radius of the Gaussian blur in the depth-of-field effect simulation is shown in the table.

Chapter 5
Conclusion

The main contribution of this thesis is a novel light field rendering method. The method outperforms the state-of-the-art methods that are analyzed in the thesis. The conducted research addresses the most crucial issues in light field rendering. A scientific hypothesis was presented, and its experimental proof showed that the proposed method for light field rendering outperforms existing approaches and solves the related issues. Additional contributions regarding the quality of rendering on 3D displays are described, as they are closely related to light fields.

The methods proposed in this thesis can be used to utilize light fields in modern industry. The rendering method allows for high-quality and real-time synthesis of a novel view from discrete light field images. Research regarding light field compression leads to the proposal of an optimal encoding and streaming scheme for light fields. It solves the excessive memory and bandwidth requirements of light field rendering methods. Hypothesis of this thesis stated that a fast and high-quality light field rendering method can be proposed. The main presented proposal focuses on optimal GPU utilization, which makes the method faster than the state of the art. The method is designed to exploit massive parallelism that allows it to densely scan the input data to find the best focus distance for each resulting pixel. This ensures mitigation of standard light field rendering out-of-focus ghosting artifacts. Hypothesis was experimentally proved as valid.

3D displays are currently not mainstream displaying technology due to various limitations such as input views alignment and a single focus distance. The proposed methods lead to automation of tasks necessary for comfortable usage of 3D displays. The methods can detect potentially good scenes from an arbitrary sequence of images or a video or detect the optimal focus level. The user study also reveals an important limitation of this technology regarding the deviation of the input data from the officially recommended guidelines.

Most of the research included in this thesis has been accepted by the scientific community, peer-reviewed, and published. The materials described in this thesis lead to a complete and usable light field rendering approach which can be utilized in industrial solutions. They can also be used as a solid basis for further research.

Future work will cover efficient compression of the 2D light field grid in the time domain. The optimal position of the I-frame can ensure a fast light field video playback. Light field rendering designed for the standard data format in a grid was concluded in this thesis. Arbitrary-positioned light field views are a topic for future research. Both optimal rendering of synthetic views and acquisition method are to be explored in depth. A comparison of several 3D viewing devices will be performed to measure the ability of users to perceive various kinds of rendering or capturing artifacts in the scene.

Chapter 5
Author's Bibliography

[1] David Bařina et al. "Comparison of light field compression methods". In: *Multimedia Tools and Applications* 81.2 (2022), pp. 2517–2528. ISSN: 1573-7721. DOI: 10.1007/s11042-021-11645-x.

[2] David Bařina et al. "Evaluation of 4D Light Field Compression Methods". In: *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Part I*. Vol. 2901. Computer Science Research Notes (CSRN). Plzeň, CZ: Union Agency, 2019, pp. 55–61. ISBN: 978-80-86943-37-4. DOI: 10.24132/CSRN.2019.2901.1.7.

[3] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "Autofocus and Out-of-Focus Artifacts Mitigation Methods for 3D Displays". In: *Visual Informatics* (Currently under review).

[4] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "Automatic 3D-Display-Friendly Scene Extraction from Video Sequences and Optimal Focusing Distance Identification". In: *Multimedia Tools and Applications* (2024), p. 29. ISSN: 1573-7721. DOI: 10.1007/s11042-024-18573-6.

[5] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "Efficient Random-Access GPU Video Decoding for Light-Field Rendering". In: *Journal of Visual Communication and Image Representation* (Currently under review).

[6] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "How Capturing Camera Trajectory Distortion Affects User Experience on Looking Glass 3D Display". In: *Multimedia Tools and Applications* 83 (Feb. 2024), pp. 20265–20287. ISSN: 1573-7721. DOI: 10.1007/s11042-023-16350-5.

[7] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "How Color Profile Affects the Visual Quality in Light Field Rendering and Novel View Synthesis". In: *Multimedia Tools and Applications* (May 2024). ISSN: 1573-7721. DOI: 10.1007/s11042-024-19396-1.

[8] Tomáš Chlubna et al. "Real-Time Light Field Video Focusing and GPU Accelerated Streaming". In: *Journal of Signal Processing Systems* (2023), pp. 1–17. DOI: 10.1007/s11265-023-01874-8.

[9] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "Real-time per-pixel focusing method for light field rendering". In: *Computational Visual Media* 2021.7 (2021), pp. 319–333. ISSN: 2096-0662. DOI: 10.1007/s41095-021-0205-0.

[10] Tomáš Chlubna, Tomáš Milet, and Pavel Zemčík. "Lightweight all-focused light field rendering". In: *Computer Vision and Image Understanding* 244 (2024), p. 104031. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2024.104031.

Chapter 5
Bibliography

[11] Amar Aggoun. "A 3D DCT Compression Algorithm For Omnidirectional Integral Images". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vol. 2. 2006. DOI: 10.1109/ICASSP.2006.1660393.

[12] Waqas Ahmad et al. "Computationally efficient light field image compression using a multiview hevc framework". In: *IEEE access* 7 (2019), pp. 143002–143014. DOI: 10.1109/ACCESS.2019.2944765.

[13] Martin Alain, Weston Aenchbacher, and Aljosa Smolic. "Interactive Light Field Tilt-Shift Refocus with Generalized Shift-and-Sum". In: *ArXiv* abs/1910.04699 (2019). DOI: 10.13140/RG.2.2.29894.01607.

[14] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. "KAZE Features". In: *Eur. Conf. on Computer Vision (ECCV)*. Fiorenze, Italy: Springer Science+Business Media, 2012. ISBN: 978-3-642-33783-3.

[15] Mosaad Alhassan et al. "Effects of Virtual Reality Head-mounted Displays on Oculomotor Functions". In: *International Journal of Ophthalmology and Visual Science* 6.1 (2021), p. 10. DOI: 10.11648/j.ijovs.20210601.12.

[16] Jurandy Almeida et al. "Robust Estimation of Camera Motion Using Optical Flow Models". In: *Advances in Visual Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 435–446. ISBN: 978-3-642-10331-5.

[17] Gustavo Alves, Fernando Pereira, and Eduardo A.B. da Silva. "Light field imaging coding: Performance assessment methodology and standards benchmarking". In: *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. 2016, pp. 1–6. DOI: 10.1109/ICMEW.2016.7574774.

[18] Yuriy Anisimov, Oliver Wasenmüller, and Didier Stricker. "Rapid Light Field Depth Estimation with Semi-Global Matching". In: *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*. 2019, pp. 109–116. DOI: 10.1109/ICCP48234.2019.8959680.

[19] Sunil Arya et al. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions". In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 891–923. ISSN: 0004-5411. DOI: 10.1145/293347.293348.

[20] Pekka Astola and Ioan Tabus. "Coding of Light Fields Using Disparity-Based Sparse Prediction". In: *IEEE Access* 7 (2019), pp. 176820–176837. DOI: 10.1109/ACCESS.2019.2957934.

[21] Vasileios Avramelos et al. "Random access prediction structures for light field video coding with MV-HEVC". In: *Multimedia Tools and Applications* 79.19 (2020), pp. 12847–12867. DOI: 10.1007/s11042-019-08605-x.

[22] Hui-Yong Bak and Seung-Bo Park. “Camera motion detection for story and multimedia information convergence”. In: *Personal and Ubiquitous Computing* 27.3 (2023), pp. 1221–1231. DOI: 10.1007/s00779-021-01585-6.

[23] Nader Bakir et al. “Light Field Image Coding Using Dual Discriminator Generative Adversarial Network And VVC Temporal Scalability”. In: *2020 IEEE International Conference on Multimedia and Expo (ICME)*. 2020, pp. 1–6. DOI: 10.1109/ICME46284.2020.9102880.

[24] Nader Bakir et al. “Light Field Image Compression Based on Convolutional Neural Networks and Linear Approximation”. In: *IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 1128–1132. DOI: 10.1109/ICIP.2018.8451597.

[25] Tibor Balogh, Péter Tamás Kovács, and Zoltán Megyesi. “Holovizio 3D display system”. In: *Proceedings of the First International Conference on Immersive Telecommunications*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2007, p. 19. ISBN: 978-1-4244-0722-4. DOI: 10.1109/3DTV.2007.4379386.

[26] Martin S Banks, Emily A Cooper, and Elise A Piazza. “Camera focal length and the perception of pictures”. In: *Ecological Psychology* 26.1-2 (2014), pp. 30–46. DOI: 10.1080/10407413.2014.877284.

[27] Christian Banz, Holger Blume, and Peter Pirsch. “Real-time semi-global matching disparity estimation on the GPU”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 514–521. DOI: 10.1109/ICCVW.2011.6130286.

[28] Brian A Barsky and Todd J Kosloff. “Algorithms for rendering depth of field effects in computer graphics”. In: *Proceedings of the 12th WSEAS international conference on Computers*. Vol. 2008. World Scientific, Engineering Academy, and Society (WSEAS). 2008. ISBN: 9789606766855.

[29] Sebastiano Battiato et al. “3D stereoscopic image pairs by depth-map generation”. In: *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.* 2004, pp. 124–131. DOI: 10.1109/TDPVT.2004.1335185.

[30] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[31] Steven Beauchemin and John Barron. “The Computation of Optical Flow”. In: *ACM Comput. Surv.* 27.3 (Sept. 1995), 433–466. ISSN: 0360-0300. DOI: 10.1145/212094.212141.

[32] James R Bergen and Edward H Adelson. “The plenoptic function and the elements of early vision”. In: *Computational models of visual processing* 1 (1991), p. 8. DOI: 10.7551/mitpress/2002.003.0004.

[33] Vasudev Bhaskaran and Konstantinos Konstantinides. “Image and video compression standards: algorithms and architectures”. In: (1997). DOI: 10.1007/978-1-4757-2358-8.

[34] Vladan Blahnik and Oliver Schindelbeck. “Smartphone imaging technology and its applications”. In: *Advanced Optical Technologies* 10.3 (2021), pp. 145–232. DOI: 10.1515/aot-2021-0023.

[35] Yuri Antonio Gonçalves Vilas Boas. “Overview of virtual reality technologies”. In: *Interactive Multimedia Conference*. Vol. 2013. 2013.

[36] Ali Borji, Dicky N Sihite, and Laurent Itti. “What stands out in a scene? A study of human explicit saliency judgment”. In: *Vision research* 91 (2013), pp. 62–77. DOI: 10.1016/j.visres.2013.07.016.

[37] Patrick Bouthemy, Marc Gelgon, and Fabrice Ganansia. “A unified approach to shot change detection and camera motion characterization”. In: *IEEE transactions on circuits and systems for video technology* 9.7 (1999), pp. 1030–1044. DOI: 10.1109/76.795057.

[38] Catarina Brites, João Ascenso, and Fernando Pereira. “Lenslet light field image coding: Classifying, reviewing and evaluating”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.1 (2020), pp. 339–354. DOI: 10.1109/TCSVT.2020.2976784.

[39] Benjamin Bross et al. “Overview of the Versatile Video Coding (VVC) Standard and its Applications”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.10 (2021), pp. 3736–3764. DOI: 10.1109/TCSVT.2021.3101953.

[40] Matthew Brown and David G. Lowe. “Automatic Panoramic Image Stitching using Invariant Features”. In: *International Journal of Computer Vision* 74.1 (Aug. 2007), pp. 59–73. ISSN: 1573-1405. DOI: 10.1007/s11263-006-0002-3.

[41] Thomas Brox et al. “High accuracy optical flow estimation based on a theory for warping”. In: *European conference on computer vision*. Springer. 2004, pp. 25–36. ISBN: 978-3-540-24673-2.

[42] Michael Broxton et al. “Immersive Light Field Video with a Layered Mesh Representation”. In: *ACM Trans. Graph.* 39.4 (Aug. 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392485.

[43] Chris Buehler et al. “Unstructured Lumigraph Rendering”. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 2001, 425–432. ISBN: 158113374X. DOI: 10.1145/383259.383309.

[44] Pierre A Buser and Michel Imbert. *Vision*. MIT press, 1992. ISBN: 978-0262023368.

[45] Emilio Camahort and Don Fussell. “A geometric study of light field representations”. In: *Technical Report TR99-35* (1999).

[46] Emilio Camahort, Apostolos Lerios, and Donald Fussell. “Uniformly sampled light fields”. In: *Rendering Techniques’ 98*. Springer, 1998, pp. 117–130. ISBN: 978-3-7091-6453-2. DOI: 10.1007/978-3-7091-6453-2_11.

[47] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890. DOI: 10.1109/TRO.2021.3075644.

[48] Sarah Cardwell. “A sense of proportion: Aspect ratio and the framing of television space”. In: *Critical Studies in Television* 10.3 (2015), pp. 83–100. DOI: 10.7227/CST.10.3.7.

[49] Jin-Xiang Chai et al. “Plenoptic Sampling”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, 307–318. ISBN: 1581132085. DOI: 10.1145/344779.344932.

[50] Hung-Chang Chang and Shang-Hong Lai. “Robust camera motion estimation and classification for video analysis”. In: *Proc SPIE* 5308 (Jan. 2004). DOI: 10.1117/12.527698.

[51] Claire Lifan Chen, C. Bhumireddy, and P. K. Darvemula. “Camera motion classification using a genetic functional-link neural network”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2343–2348 vol.3. DOI: 10.1109/IROS.2004.1389759.

[52] Jie Chen, Junhui Hou, and Lap-Pui Chau. “Light Field Compression With Disparity-Guided Sparse Coding Based on Structural Key Views”. In: *IEEE Transactions on Image Processing* 27.1 (2018), pp. 314–324. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2750413.

[53] Xuechao Chen et al. “Efficiently reconstruct light field probes from light-G-buffers”. In: *Computers & Graphics* 94 (2021), pp. 144–151. ISSN: 0097-8493. DOI: 10.1016/j.cag.2020.12.003.

[54] Yang Chen, Martin Alain, and Aljosa Smolic. “Fast and accurate optical flow based depth map estimation from light fields”. In: *Irish Machine Vision and Image Processing Conference (IMVIP)*. 2017. DOI: arXiv:2008.04673.

[55] Yang Chen, Martin Alain, and Aljosa Smolic. *View Sub-sampling and Reconstruction for Efficient Light Field Compression*. 2022. arXiv: 2208.06464 [eess.IV].

[56] Eun Joung Cho and Kwan Min Lee. “Effects of 3D displays: A comparison between shuttered and polarized displays”. In: *Displays* 34.5 (2013), pp. 353–358. ISSN: 0141-9382. DOI: 10.1016/j.displa.2013.09.003.

[57] Myungsub Choi et al. “Scene-Adaptive Video Frame Interpolation via Meta-Learning”. In: *CVPR*. 2020. DOI: 10.48550/arXiv.2004.00779.

[58] Ravi Choudhary, Vidhi Goel, and Gaurav Meena. “Survey Paper: Image Quality Assessment”. In: *SSRN Electronic Journal* (Jan. 2019). DOI: 10.2139/ssrn.3356307.

[59] Aleksandra Chuchvara, Attila Barsi, and Atanas Gotchev. “Fast and Accurate Depth Estimation From Sparse Light Fields”. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 2492–2506. DOI: 10.1109/TIP.2019.2959233.

[60] Duolikun Danier, Fan Zhang, and David Bull. “ST-MFNet: A Spatio-Temporal Multi-Flow Network for Frame Interpolation”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2022, pp. 3511–3521. DOI: 10.1109/CVPR52688.2022.00351.

[61] Donald G. Dansereau, Oscar Pizarro, and Stefan B. Williams. “Linear Volumetric Focus for Light Field Cameras”. In: *ACM Trans. Graph.* 34.2 (Mar. 2015). ISSN: 0730-0301. DOI: 10.1145/2665074. URL: https://doi.org/10.1145/2665074.

[62] Paul E. Debevec, Yizhou Yu, and George Borshukov. “Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping”. In: *Rendering Techniques*. 1998. DOI: 10.1007/978-3-7091-6453-2_10.

[63] Frederik Dekking et al. *A Modern Introduction to Probability and Statistics, Understanding Why and How*. Springer, Jan. 2005. ISBN: 978-1-85233-896-1. DOI: 10.1007/1-84628-168-7.

[64] Ibrahim Delibasoglu et al. “Motion detection in moving camera videos using background modeling and FlowNet”. In: *Journal of Visual Communication and Image Representation* 88 (2022), p. 103616. ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2022.103616.

[65] Robert William Ditchburn. *Light*. Dover Publications, 2011. ISBN: 978-0486666679.

[66] Jiong Dong, Kaoru Ota, and Mianxiong Dong. “Video Frame Interpolation: A Comprehensive Survey”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 19.2s (May 2023). ISSN: 1551-6857. DOI: 10.1145/3556544.

[67] Tuochuan Dong et al. “A new diagnostic accuracy measure and cut-point selection criterion”. In: *Statistical methods in medical research* 26 (Oct. 2015). DOI: 10.1177/0962280215611631.

[68] Joost van Dongen. “Interior mapping”. In: *CGI 2008 Conference Proceedings*. 2008.

[69] Kim Donghyun and Kwanghoon Sohn. “Visual Fatigue Prediction for Stereoscopic Image”. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 21 (Mar. 2011), pp. 231–236. DOI: 10.1109/TCSVT.2011.2106275.

[70] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’14. Montreal, Canada: MIT Press, 2014, 2366–2374.

[71] Fatima El Jamiy and Ronald Marsh. “Survey on depth perception in head mounted displays: distance estimation in virtual reality, augmented reality, and mixed reality”. In: *IET Image Processing* 13.5 (2019), pp. 707–712. DOI: 10.1049/iet-ipr.2018.5920.

[72] Seif Allah El Mesloul Nasri et al. “Multiview Video Coding: A Comparative Study Between MVC and MV-HEVC”. In: *Recent Trends in Computer Applications*. Ed. by Jihad Mohamad Alja’am, Abdulmotaleb El Saddik, and Abdul Hamid Sadka. Cham: Springer International Publishing, 2018, pp. 99–112. ISBN: 978-3-319-89914-5. DOI: 10.1007/978-3-319-89914-5_7.

[73] James H Elder and Steven W Zucker. “Local scale control for edge detection and blur estimation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 20.7 (1998), pp. 699–716. DOI: 10.1109/34.689301.

[74] Okan Erat et al. “Real-Time View Planning for Unstructured Lumigraph Modeling”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.11 (2019), pp. 3063–3072. DOI: 10.1109/TVCG.2019.2932237.

[75] S. M. Ali Eslami et al. “Neural scene representation and rendering”. In: *Science* 360.6394 (2018), pp. 1204–1210. ISSN: 0036-8075. DOI: 10.1126/science.aar6170.

[76] Michael Faraday. “LIV. Thoughts on ray-vibrations”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 28.188 (1846), pp. 345–350. DOI: 10.1080/14786444608645431.

[77] Gunnar Farnebäck. “Fast and accurate motion estimation using orientation tensors and parametric motion models”. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 1. 2000, 135–139 vol.1. DOI: 10.1109/ICPR.2000.905291.

[78] David Fattal. “LEIA: The Lume Pad: 3D Lightfield at your fingertips”. In: *SPIE AVR21 Industry Talks II*. Ed. by Conference Chair. Vol. 11764. International Society for Optics and Photonics. SPIE, 2021, p. 1176415. DOI: 10.1117/12.2597482.

[79] Jocelyn Faubert. “Motion parallax, stereoscopy, and the perception of depth: Practical and theoretical issues”. In: *Three-Dimensional Video and Display: Devices and Systems: A Critical Review*. Vol. 10298. SPIE. 2001, pp. 168–191. DOI: 10.1117/12.419794.

[80] Steven H Ferris. “Motion parallax and absolute distance.” In: *Journal of experimental psychology* 95.2 (1972), p. 258. DOI: 10.1037/h0033605.

[81] Shawn Frayne, Tung Yiu Fok, and Shiu Pong Lee. *Superstereoscopic display with enhanced off-angle separation*. US Patent 10,298,921. May 2019.

[82] Shawn Frayne et al. *Advanced retroreflecting aerial displays*. US Patent App. 10/012,841. July 2018.

[83] Keren Fu et al. “Light field salient object detection: A review and benchmark”. In: *Computational Visual Media* (2022), pp. 1–26. DOI: 10.1007/s41095-021-0256-2.

[84] Zhongpai Gao et al. “Correcting geometric distortions in stereoscopic 3D imaging”. In: *PloS one* 13.10 (2018), e0205032. DOI: 10.1371/journal.pone.0205032.

[85] Andreas Geiger, Julius Ziegler, and Christoph Stiller. “StereoScan: Dense 3d reconstruction in real-time”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 963–968. DOI: 10.1109/IVS.2011.5940405.

[86] Yuliang Geng et al. “A Robust and Hierarchical Approach for Camera Motion Classification”. In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Dit-Yan Yeung et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 340–348. ISBN: 978-3-540-37241-7.

[87] Todor Georgiev et al. “Lytro camera technology: theory, algorithms, performance analysis”. In: *Multimedia Content and Mobile Devices*. Vol. 8667. International Society for Optics and Photonics. 2013, 86671J. DOI: 10.1117/12.2013581.

[88] Andreī Gershun. “The light field”. In: *Journal of Mathematics and Physics* 18.1-4 (1939), pp. 51–151. DOI: 10.1002/sapm193918151.

[89] Eleanor J Gibson et al. “Motion parallax as a determinant of perceived depth.” In: *Journal of experimental psychology* 58.1 (1959), p. 40. DOI: 10.1037/h0043883.

[90] Steven J. Gortler et al. “The Lumigraph”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, 1996, 43–54. ISBN: 0897917464. DOI: 10.1145/237170.237200.

[91] Jason Gregory. *Game engine architecture*. AK Peters/CRC Press, 2018. ISBN: 978-1466560017.

[92] Ke Gu et al. "Multiscale Natural Scene Statistical Analysis for No-Reference Quality Evaluation of DIBR-Synthesized Views". In: *IEEE Transactions on Broadcasting* 66.1 (2020), pp. 127–139. DOI: 10.1109/TBC.2019.2906768.

[93] Xiaodong Gu et al. "Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2492–2501. DOI: 10.1109/CVPR42600.2020.00257.

[94] Bob D Guenther. *Modern optics*. OUP Oxford, 2015. ISBN: 0198738773.

[95] Mickael Guironnet, Denis Pellerin, and Michèle Rombaut. "Camera motion classification based on Transferable Belief Model". In: *2006 14th European Signal Processing Conference*. 2006, pp. 1–5.

[96] Heng Guo et al. "View-consistent meshflow for stereoscopic video stabilization". In: *IEEE Transactions on Computational Imaging* 4.4 (2018), pp. 573–584. DOI: 10.1109/TCI.2018.2866227.

[97] Jie Guo et al. "Efficient Light Probes for Real-Time Global Illumination". In: *ACM Trans. Graph.* 41.6 (Nov. 2022). ISSN: 0730-0301. DOI: 10.1145/3550454.3555452.

[98] Shantanu Sen Gupta, Partha Pratim Banik, and Ki-Doo Kim. "Study on the Log-encoding System for a Camera Image Sensor". In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. 2019, pp. 1047–1049. DOI: 10.1109/ICTC46691.2019.8939972.

[99] Christopher Hahne et al. "Refocusing distance of a standard plenoptic camera". In: *Opt. Express* 24.19 (Sept. 2016), pp. 21521–21540. DOI: 10.1364/OE.24.021521.

[100] Saad Hikmat Haji and Adnan Mohsin Abdulazeez. "Comparison of optimization techniques based on gradient descent algorithm: A review". In: *PalArch's Journal of Archaeology of Egypt/Egyptology* 18.4 (2021), pp. 2715–2743. ISSN: 1567214X.

[101] Matt Hall. "Smooth operator: Smoothing seismic interpretations and attributes". In: *The Leading Edge* 26.1 (2007), pp. 16–20. DOI: 10.1190/1.2431821.

[102] Rostam Affendi Hamzah and Haidi Ibrahim. "Literature survey on stereo vision disparity map algorithms". In: *Journal of Sensors* 2016 (2016). DOI: 10.1155/2016/8742920.

[103] Haixu Han, Jin Xin, and Qionghai Dai. "Plenoptic Image Compression via Simplified Subaperture Projection". In: *Advances in Multimedia Information Processing (PCM)*. 2018, pp. 274–284. ISBN: 978-3-030-00767-6.

[104] Xianfeng Han, Hamid Laga, and Mohammed Bennamoun. "Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2019.2954885.

[105] Muhammad Hasan et al. "CAMHID: Camera Motion Histogram Descriptor and Its Application to Cinematographic Shot Classification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.10 (2014), pp. 1682–1695. DOI: 10.1109/TCSVT.2014.2345933.

[106] Eisa Hedayati, Timothy C. Havens, and Jeremy P. Bos. “Light Field Compression by Residual CNN-Assisted JPEG”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–9. DOI: 10.1109/IJCNN52387.2021.9534210.

[107] Benno Heigl et al. “Plenoptic modeling and rendering from image sequences taken by a hand-held camera”. In: *Mustererkennung 1999: 21. DAGM-Symposium Bonn, 15.–17. September 1999*. Springer. 1999, pp. 94–101. ISBN: 978-3-642-60243-6. DOI: 10.1007/978-3-642-60243-6_11.

[108] Robert Held and Martin Banks. “Misperceptions in Stereoscopic Displays: A Vision Science Perspective”. In: *ACM transactions on graphics* 2008 (Jan. 2008), pp. 23–32. DOI: 10.1145/1394281.1394285.

[109] Thorsten Herfet, Tobias Lange, and Harini Priyadarshini Hariharan. “Enabling Multiview-and Light Field-Video for Veridical Visual Experiences”. In: *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. IEEE. 2018, pp. 1705–1709. DOI: 10.1109/CompComm.2018.8780991.

[110] Rogerio Seiji Higa et al. “Plenoptic image compression comparison between JPEG, JPEG2000 and SPIHT”. In: *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)* 3.6 (2013). ISSN: 1925-2676.

[111] Matthew Hirsch et al. “Tensor Displays”. In: *ACM SIGGRAPH 2012 Emerging Technologies*. SIGGRAPH ’12. Los Angeles, California: Association for Computing Machinery, 2012. ISBN: 9781450316804. DOI: 10.1145/2343456.2343480.

[112] Heiko Hirschmuller. “Stereo Processing by Semiglobal Matching and Mutual Information”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (Feb. 2008), pp. 328–341. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2007.1166.

[113] Katrin Honauer et al. “A dataset and evaluation methodology for depth estimation on 4D light fields”. In: *Asian conference on computer vision*. Springer. 2017, pp. 19–34. ISBN: 978-3-319-54187-7. DOI: 10.1007/978-3-319-54187-7_2.

[114] Berthold Horn and Brian Schunck. “Determining optical flow”. In: *Artificial Intelligence* 17.1 (1981), pp. 185–203. ISSN: 0004-3702. DOI: 10.1016/0004-3702(81)90024-2.

[115] Daniel Reiter Horn and Billy Chen. “LightShop: Interactive Light Field Manipulation and Rendering”. In: *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D ’07. Seattle, Washington: Association for Computing Machinery, 2007, 121–128. ISBN: 9781595936288. DOI: 10.1145/1230100.1230121.

[116] Junhui Hou, Jie Chen, and Lap-Pui Chau. “Light Field Image Compression Based on Bi-Level View Compensation With Rate-Distortion Optimization”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.2 (2019), pp. 517–530. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2018.2802943.

[117] Noor A Ibraheem et al. “Understanding color models: a review”. In: *ARPN Journal of science and technology* 2.3 (2012), pp. 265–275. ISSN: 2225-7217.

[118] Insung Ihm, Sanghoon Park, and Rae Kyoung Lee. “Rendering of spherical light fields”. In: *Proceedings The Fifth Pacific Conference on Computer Graphics and Applications*. 1997, pp. 59–68. DOI: 10.1109/PCCGA.1997.626172.

[119] Wijnand A IJsselsteijn, Huib de Ridder, and Joyce Vliegen. “Effects of stereoscopic filming parameters and display duration on the subjective assessment of eye strain”. In: *Stereoscopic Displays and Virtual Reality Systems VII*. Vol. 3957. SPIE. 2000, pp. 12–22. DOI: 10.1117/12.384448.

[120] Wijnand A IJsselsteijn, Huib de Ridder, and Joyce Vliegen. “Subjective evaluation of stereoscopic images: effects of camera parameters and display duration”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 10.2 (2000), pp. 225–233. DOI: 10.1109/76.825722.

[121] Eddy Ilg et al. “Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 614–630. ISBN: 978-3-030-01258-8. DOI: 10.1007/978-3-030-01258-8_38.

[122] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. “Dynamically Reparameterized Light Fields”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, 297–306. ISBN: 1581132085. DOI: 10.1145/344779.344929.

[123] Aaron Isaksen, Leonard McMillan, and Steven J Gortler. “Dynamically reparameterized light fields”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 297–306. DOI: 10.1145/344779.344929.

[124] Hae-Gon Jeon et al. “Accurate depth map estimation from a lenslet light field camera”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1547–1555. DOI: 10.1109/CVPR.2015.7298762.

[125] Chuanmin Jia et al. “Light Field Image Compression Using Generative Adversarial Network-Based View Synthesis”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.1 (2019), pp. 177–189. DOI: 10.1109/JETCAS.2018.2886642.

[126] Shi Jianbo and Carlo Tomasi. “Good features to track”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[127] Huaizu Jiang et al. “Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation”. In: (2017). DOI: 10.48550/ARXIV.1712.00080.

[128] Xiaoran Jiang, Mikaël Le Pendu, and Christine Guillemot. “Depth Estimation with Occlusion Handling from a Sparse Set of Light Field Views”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 634–638. DOI: 10.1109/ICIP.2018.8451466.

[129] Xiaoran Jiang, Jinglei Shi, and Christine Guillemot. “An Untrained Neural Network Prior for Light Field Compression”. In: *IEEE Transactions on Image Processing* 31 (2022), pp. 6922–6936. DOI: 10.1109/TIP.2022.3217374.

[130] Xiaoran Jiang et al. “Light Field Compression With Homography-Based Low-Rank Approximation”. In: *IEEE Journal of Selected Topics in Signal Processing* 11.7 (2017), pp. 1132–1145. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2017.2747078.

[131] Andrew Jones et al. “Rendering for an Interactive 360° Light Field Display”. In: *ACM Trans. Graph.* 26.3 (July 2007), 40–es. ISSN: 0730-0301. DOI: 10.1145/1276377.1276427.

[132] Cheolkon Jung and Shuai Wang. “Visual comfort assessment in stereoscopic 3D images using salient object disparity”. In: *Electronics Letters* 51.6 (2015), pp. 482–484. DOI: 10.1049/el.2014.3913.

[133] Yong Ju Jung et al. “Visual Importance- and Discomfort Region-Selective Low-Pass Filtering for Reducing Visual Discomfort in Stereoscopic Displays”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.8 (2013), pp. 1408–1421. DOI: 10.1109/TCSVT.2013.2244796.

[134] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. “Learning-Based View Synthesis for Light Field Cameras”. In: *ACM Trans. Graph.* 35.6 (Nov. 2016). ISSN: 0730-0301. DOI: 10.1145/2980179.2980251.

[135] Tarun Kalluri et al. “FLAVR: Flow-Agnostic Video Representations for Fast Frame Interpolation”. In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, pp. 2070–2081. DOI: 10.1109/WACV56688.2023.00211.

[136] Zhizhong Kang et al. “A review of techniques for 3d reconstruction of indoor environments”. In: *ISPRS International Journal of Geo-Information* 9.5 (2020), p. 330. DOI: 10.3390/ijgi9050330.

[137] Ketan Kapse. “An Overview of Current Deep Learned Rendering Technologies”. In: *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Vol. 1. 2021, pp. 1404–1409. DOI: 10.1109/ICACCS51430.2021.9441822.

[138] Edyta Karpicka and Peter Alan Howarth. “Heterophoria adaptation during the viewing of 3D stereoscopic stimuli”. In: *Ophthalmic and Physiological Optics* 33.5 (2013), pp. 604–610. DOI: 10.1111/opo.12081.

[139] Masaki Kawase. “Frame Buffer Postprocessing Effects in DOUBLE-STEAL (Wrechless)”. In: *Game Developers Conference 2003, 3*. 2003.

[140] Sean Frederick Keane et al. *Volumetric 3D display*. US Patent 10,401,636. Sept. 2019.

[141] Joseph Khoury, Mahsa T Pourazad, and Panos Nasiopoulos. “A new prediction structure for efficient MV-HEVC based light field video compression”. In: *2019 International conference on computing, networking and communications (ICNC)*. IEEE. 2019, pp. 588–591. DOI: 10.1109/ICCNC.2019.8685526.

[142] Jyrki Kimmel. *Diffractive backlight structure*. US Patent App. 13/128,628. May 2012.

[143] Vladimir Kolmogorov and Ramin Zabih. “Multi-camera scene reconstruction via graph cuts”. In: *Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part III 7*. Springer. 2002, pp. 82–96. DOI: 10.1007/3-540-47977-5_6.

[144] Lingtong Kong et al. “IFRNet: Intermediate Feature Refine Network for Efficient Frame Interpolation”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 1959–1968. DOI: 10.1109/CVPR52688.2022.00201.

[145] Babis Koniaris et al. “Real-Time Rendering with Compressed Animated Light Fields”. In: *Proceedings of the 43rd Graphics Interface Conference*. GI ’17. Edmonton, Alberta, Canada: Canadian Human-Computer Communications Society, 2017, 33–40. ISBN: 9780994786821.

[146] Péter Tamás Kovács et al. “Architectures and codecs for real-time light field streaming”. In: *Journal of Imaging Science and Technology* 61.1 (2017), pp. 10403–1. DOI: 10.2352/J.ImagingSci.Technol.2017.61.1.010403.

[147] Ernst Kruijff, J Edward Swan, and Steven Feiner. “Perceptual issues in augmented reality revisited”. In: *2010 IEEE International Symposium on Mixed and Augmented Reality*. IEEE. 2010, pp. 3–12. DOI: 10.1109/ISMAR.2010.5643530.

[148] Akira Kubota et al. “All-focused light field rendering”. In: *Eurographics Workshop on Rendering*. Ed. by Alexander Keller and Henrik Wann Jensen. The Eurographics Association, 2004. ISBN: 3-905673-12-6. DOI: 10.2312/EGWR/EGSR04/235-242.

[149] Heljä Kukkonen et al. “Michelson contrast, RMS contrast and energy of various spatial stimuli at threshold”. In: *Vision Research* 33.10 (1993), pp. 1431–1436. ISSN: 0042-6989. DOI: 10.1016/0042-6989(93)90049-3.

[150] Michiyoshi. Kuwahara et al. “Processing of RI-Angiocardiographic Images”. In: *Digital Processing of Biomedical Images*. Ed. by K. Preston and M. Onoe. Boston, MA: Springer US, 1976, pp. 187–202. ISBN: 978-1-4684-0769-3. DOI: 10.1007/978-1-4684-0769-3_13.

[151] Marc TM Lambooij, Wijnand A IJsselsteijn, and Ingrid Heynderickx. “Visual discomfort in stereoscopic displays: a review”. In: *Stereoscopic Displays and Virtual Reality Systems XIV* 6490 (2007), pp. 183–195. DOI: 10.1117/12.705527.

[152] Mikaël Le Pendu, Christine Guillemot, and Aljosa Smolic. “A Fourier Disparity Layer Representation for Light Fields”. In: *IEEE Transactions on Image Processing* 28.11 (2019), pp. 5740–5753. DOI: 10.1109/TIP.2019.2922099.

[153] Sangkeun Lee and Monson Hayes. “Real-time camera motion classification for content-based indexing and retrieval using templates”. In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. 2002, pp. IV–3664–IV–3667. DOI: 10.1109/ICASSP.2002.5745450.

[154] Seungho Lee, Hyunmin Jung, and Chae Eun Rhee. “Data Orchestration for Accelerating GPU-Based Light Field Rendering Aiming at a Wide Virtual Space”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 32.6 (2022), pp. 3575–3586. DOI: 10.1109/TCSVT.2021.3121010.

[155] Sungkil Lee, Younguk Kim, and Elmar Eisemann. “Iterative Depth Warping”. In: *ACM Trans. Graph.* 37.5 (Oct. 2018). ISSN: 0730-0301. DOI: 10.1145/3190859.

[156] Titus Leistner et al. “Learning to Think Outside the Box: Wide-Baseline Light Field Depth Estimation with EPI-Shift”. In: *2019 International Conference on 3D Vision (3DV)*. 2019, pp. 249–257. DOI: 10.1109/3DV.2019.00036.

[157] Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov. “Deep Image Prior”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9446–9454. DOI: 10.1109/CVPR.2018.00984.

[158] Marc Levoy and Pat Hanrahan. “Light Field Rendering”. In: SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, 1996, 31–42. ISBN: 0897917464. DOI: 10.1145/237170.237199.

[159] Marc Levoy et al. “Light field microscopy”. In: *ACM SIGGRAPH 2006 Papers*. SIGGRAPH ’06. New York, NY, USA: Association for Computing Machinery, 2006, 924–934. ISBN: 1595933646. DOI: 10.1145/1179352.1141976. URL: https://doi.org/10.1145/1179352.1141976.

[160] Chen Li and Xu Zhang. “High dynamic range and all-focus image from light field”. In: *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 2015, pp. 7–12. DOI: 10.1109/ICCIS.2015.7274539.

[161] Jiahao Li, Bin Li, and Yan Lu. “Neural Video Compression with Diverse Contexts”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, Canada, June 18-22, 2023*. 2023. DOI: 10.1109/CVPR52729.2023.02166.

[162] Li Li and Zhu Li. “Light Field Image Compression”. In: *Handbook of Dynamic Data Driven Applications Systems*. Ed. by Erik Blasch, Sai Ravela, and Alex Aved. Springer, 2018, pp. 547–570. ISBN: 978-3-319-95504-9. DOI: 10.1007/978-3-319-95504-9_24.

[163] Li Li et al. “Pseudo Sequence Based 2-D Hierarchical Coding Structure for Light-Field Image Compression”. In: *Data Compression Conference (DCC)*. 2017, pp. 131–140. DOI: 10.1109/DCC.2017.10.

[164] Qun Li and Dan Schonfeld. “General stereoscopic distortion rectification due to arbitrary viewer motion in binocular stereoscopic display”. In: vol. 9011. Feb. 2014. DOI: 10.1117/12.2038282.

[165] Yun Li et al. “Coding of Focused Plenoptic Contents by Displacement Intra Prediction”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.7 (2016), pp. 1308–1319. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2015.2450333.

[166] Yun Li et al. “Efficient intra prediction scheme for light field image compression”. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 2014, pp. 539–543. DOI: 10.1109/ICASSP.2014.6853654.

[167] Haiting Lin et al. “Depth Recovery from Light Field Using Focal Stack Symmetry”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 3451–3459. DOI: 10.1109/ICCV.2015.394.

[168] Zhouchen Lin and Heung-Yeung Shum. “A Geometric Analysis of Light Field Rendering”. In: *Int. J. Comput. Vision* 58.2 (June 2004), 121–138. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000015916.91741.27.

[169] Wilfried Linder. *Digital photogrammetry*. Springer, 2009. ISBN: 978-3-662-50463-5. DOI: 10.1007/978-3-662-50463-5.

[170] Ling-Yu Duan et al. “Nonparametric motion characterization for robust classification of camera motion patterns”. In: *IEEE Transactions on Multimedia* 8.2 (2006), pp. 323–340. DOI: 10.1109/TMM.2005.864344.

[171] Deyang Liu et al. “Multi-Stream Dense View Reconstruction Network for Light Field Image Compression”. In: *IEEE Transactions on Multimedia* (2022), pp. 1–1. DOI: 10.1109/TMM.2022.3175023.

[172] Feng Liu, Yuzhen Niu, and Hailin Jin. “Joint subspace stabilization for stereoscopic video”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 73–80. DOI: 10.1109/ICCV.2013.16.

[173] Lina Liu, Ru Zhang, and Ling Fan. “Camera motion classification based on SVM”. In: *2010 3rd International Congress on Image and Signal Processing*. Vol. 1. 2010, pp. 392–394. DOI: 10.1109/CISP.2010.5648012.

[174] Yi Liu et al. “Subjective assessment on visual fatigue versus stereoscopic disparities”. In: *Journal of the Society for Information Display* 29.6 (2021), pp. 497–504. DOI: 10.1002/jsid.991.

[175] Mark A. Livingston et al. “Vertical Vergence Calibration for Augmented Reality Displays”. In: *IEEE Virtual Reality Conference (VR 2006)*. 2006, pp. 287–288. DOI: 10.1109/VR.2006.142.

[176] Martin Loesdau, Sébastien Chabrier, and Alban Gabillon. “Hue and saturation in the RGB color space”. In: *International conference on image and signal processing*. Springer. 2014, pp. 203–212. ISBN: 978-3-319-07998-1. DOI: 10.1007/978-3-319-07998-1_23.

[177] Bruce Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)”. In: *IJCAI’81: Proceedings of the 7th international joint conference on Artificial intelligence*. Vol. 81. Apr. 1981.

[178] David Luengo et al. “A survey of Monte Carlo methods for parameter estimation”. In: *EURASIP Journal on Advances in Signal Processing* 2020.1 (2020), pp. 1–62. DOI: 10.1186/s13634-020-00675-6.

[179] Marcus Magnor and Bernd Girod. “Data compression for light-field rendering”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 10.3 (2000), pp. 338–343. DOI: 10.1109/76.836278.

[180] Morgan McGuire et al. “Real-Time Global Illumination Using Precomputed Light Field Probes”. In: *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’17. San Francisco, California: Association for Computing Machinery, 2017. ISBN: 9781450348867. DOI: 10.1145/3023368.3023378.

[181] Nusrat Mehajabin et al. “An efficient random access light field video compression utilizing diagonal inter-view prediction”. In: *2019 IEEE International conference on image processing (ICIP)*. IEEE. 2019, pp. 3567–3570. DOI: 10.1109/ICIP.2019.8803668.

[182] Mostafa Mehrabi et al. “Making 3D work: a classification of visual depth cues, 3D display technologies and their applications”. In: Jan. 2013, pp. 91–100. ISBN: 9781921770241.

[183] Moritz Menze, Christian Heipke, and Andreas Geiger. "Object Scene Flow". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 140 (2018). Geospatial Computer Vision, pp. 60–76. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2017.09.013.

[184] Ehsan Miandji, Saghi Hajisharif, and Jonas Unger. "A Unified Framework for Compression and Compressed Sensing of Light Fields and Light Field Videos". In: *ACM Trans. Graph.* 38.3 (May 2019). ISSN: 0730-0301. DOI: 10.1145/3269980.

[185] Sanjida Sharmin Mohona, Laurie M Wilcox, and Robert S Allison. "Subjective assessment of display stream compression for stereoscopic imagery". In: *Journal of the Society for Information Display* 29.8 (2021), pp. 591–607. DOI: https://doi.org/10.1002/jsid.1002.

[186] Christian Moller and Adrian Travis. "Correcting interperspective aliasing in autostereoscopic displays". In: *IEEE Transactions on Visualization and Computer Graphics* 11.2 (2005), pp. 228–236. DOI: 10.1109/TVCG.2005.28.

[187] Ricardo J. S. Monteiro et al. "Light Field Image Coding Based on Hybrid Data Representation". In: *IEEE Access* 8 (2020), pp. 115728–115744. DOI: 10.1109/ACCESS.2020.3004625.

[188] Theo Moons, Luc Van Gool, and Maarten Vergauwen. "3D Reconstruction from Multiple Images Part 1: Principles". In: *Found. Trends. Comput. Graph. Vis.* 4.4 (Apr. 2010), 287–404. ISSN: 1572-2740. DOI: 10.1561/0600000007.

[189] Guillaume Moreau. "Visual Immersion Issues in Virtual Reality: A Survey". In: *Proceedings: 26th SIBGRAPI - Conference on Graphics, Patterns and Images Tutorials, SIBGRAPI-T 2013* (Nov. 2014), pp. 6–14. DOI: 10.1109/SIBGRAPI-T.2013.9.

[190] Thomas Müller et al. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127.

[191] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), 1147–1163. ISSN: 1941-0468. DOI: 10.1109/tro.2015.2463671.

[192] Raúl Mur-Artal and Juan D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[193] Masaki Naito et al. "Camera Motion Detection using Video Mosaicing". In: *2006 IEEE International Conference on Multimedia and Expo*. 2006, pp. 1741–1744. DOI: 10.1109/ICME.2006.262887.

[194] Alessandro Neri, Marco Carli, and Federica Battisti. "A multi-resolution approach to depth field estimation in dense image arrays". In: *2015 IEEE International Conference on Image Processing (ICIP)*. 2015, pp. 3358–3362. DOI: 10.1109/ICIP.2015.7351426.

[195] Ren Ng et al. "Light field photography with a hand-held plenoptic camera". In: *Computer Science Technical Report CSTR* 2.11 (2005), pp. 1–11.

[196] Lixia Ni et al. “Unsupervised Dense Light Field Reconstruction with Occlusion Awareness”. In: *Computer Graphics Forum* 38.7 (2019), pp. 425–436. ISSN: 0167-7055. DOI: 10.1111/cgf.13849.

[197] Simon Niklaus and Feng Liu. “Softmax Splatting for Video Frame Interpolation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2020, pp. 5436–5445. DOI: 10.1109/CVPR42600.2020.00548.

[198] Justin O’Brien and Alan Johnston. “When texture takes precedence over motion in depth perception”. In: *Perception* 29.4 (2000), pp. 437–452. DOI: 10.1068/p2955.

[199] Jean-Marc Odobez and Patrick Bouthemy. “Robust Multiresolution Estimation of Parametric Motion Models”. In: *Journal of Visual Communication and Image Representation* 6.4 (1995), pp. 348–365. ISSN: 1047-3203. DOI: 10.1006/jvci.1995.1029.

[200] Takanori Okoshi. *Three-Dimensional Imaging Techniques*. Atara Press, 2011. ISBN: 9780982225141.

[201] Ryan S. Overbeck et al. “The Making of Welcome to Light Fields VR”. In: *ACM SIGGRAPH 2018 Talks*. SIGGRAPH ’18. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2018. ISBN: 9781450358200. DOI: 10.1145/3214745.3214811.

[202] Roberta de Carvalho Nobre Palau et al. “Modern Video Coding: Methods, Challenges and Systems”. In: *Journal of Integrated Circuits and Systems* 16.2 (2021), pp. 1–12. DOI: 10.29292/jics.v16i2.503.

[203] Luca Palmieri, Reinhard Koch, and Ron Op Het Veld. “The Plenoptic 2.0 Toolbox: Benchmarking of Depth Estimation Methods for MLA-Based Focused Plenoptic Cameras”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. Feb. 2018, pp. 649–653. DOI: 10.1109/ICIP.2018.8451073.

[204] Junheum Park, Jintae Kim, and Chang-Su Kim. “BiFormer: Learning Bilateral Motion Estimation via Bilateral Transformer for 4K Video Frame Interpolation”. In: *Computer Vision and Pattern Recognition*. 2023. DOI: 10.1109/CVPR52729.2023.00157.

[205] Sang-Cheol Park, Hyoung-Suk Lee, and Seong-Whan Lee. “Qualitative estimation of camera motion parameters from the linear composition of optical flow”. In: *Pattern Recognition* 37.4 (2004). Agent Based Computer Vision, pp. 767–779. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2003.07.012.

[206] Jose Luis Pech Pacheco et al. “Diatom autofocusing in brightfield microscopy: a comparative study”. In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 3. 2000, 314–317 vol.3. DOI: 10.1109/ICPR.2000.903548.

[207] Jiayong Peng et al. “Unsupervised Depth Estimation from Light Field Using a Convolutional Neural Network”. In: *2018 International Conference on 3D Vision (3DV)*. Sept. 2018, pp. 295–303. DOI: 10.1109/3DV.2018.00042.

[208] Cristian Perra and Daniele Giusto. “JPEG 2000 compression of unfocused light field images based on lenslet array slicing”. In: *IEEE International Conference on Consumer Electronics (ICCE)*. 2017, pp. 27–28. DOI: 10.1109/ICCE.2017.7889217.

[209] Cristian Perra, Francesca Murgia, and Daniele Giusto. “An analysis of 3D point cloud reconstruction from light field images”. In: *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 2016, pp. 1–6. DOI: 10.1109/IPTA.2016.7821011.

[210] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. “Analysis of focus measure operators for shape-from-focus”. In: *Pattern Recognition* 46.5 (2013), pp. 1415–1432. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2012.11.011.

[211] Ulrich Perwass and Christian Perwass. *Digital imaging system, plenoptic optical device and image data processing method*. US Patent 8,619,177. Dec. 2013.

[212] Srihar Pratapa and Dinesh Manocha. “HMLFC: Hierarchical Motion-Compensated Light Field Compression for Interactive Rendering”. In: *arXiv preprint arXiv:1902.09396* (2019). DOI: 10.48550/arXiv.1902.09396.

[213] Amal Punchihewa, Takayuki Hamamoto, and Takahiro Kojima. “From a review of HDR sensing and tone compression to a novel imaging approach”. In: *2011 Fifth International Conference on Sensing Technology*. 2011, pp. 40–46. DOI: 10.1109/ICSensT.2011.6137010.

[214] Anyi Rao et al. “A Unified Framework for Shot Type Classification Based on Subject Centric Lens”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 17–34. ISBN: 978-3-030-58621-8.

[215] Joshitha Ravishankar, Sally Khaidem, and Mansi Sharma. “A Data-Driven Approach Based on Dynamic Mode Decomposition for Efficient Encoding of Dynamic Light Fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 3446–3452. DOI: 10.1109/CVPRW59228.2023.00347.

[216] Fitsum Reda et al. “FILM: Frame Interpolation for Large Motion”. In: *ECCV 2022* (2022). DOI: 10.48550/arXiv.2202.04901.

[217] Fitsum Reda et al. *Tensorflow 2 Implementation of „FILM: Frame Interpolation for Large Motion“*. https://github.com/google-research/frame-interpolation. 2022.

[218] Stephan Reichelt et al. “Depth cues in human visual perception and their realization in 3D displays”. In: *Three-Dimensional Imaging, Visualization, and Display 2010 and Display Technologies and Applications for Defense, Security, and Avionics IV*. Ed. by Bahram Javidi et al. Vol. 7690. International Society for Optics and Photonics. SPIE, 2010, pp. 92 –103. DOI: 10.1117/12.850094.

[219] David Rempel et al. “The Effects of Visual Display Distance on Eye Accommodation, Head Posture, and Vision and Neck Symptoms”. In: *Human factors* 49 (Nov. 2007), pp. 830–8. DOI: 10.1518/001872007X230208.

[220] Martin Rerabek and Touradj Ebrahimi. “New light field image dataset”. In: *8th International Conference on Quality of Multimedia Experience (QoMEX)*. CONF. 2016.

[221] Paul Rosenthal and Lars Linsen. “Image-space point cloud rendering”. In: *Proceedings of Computer Graphics International*. 2008, pp. 136–143.

[222] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. Institute of Electrical and Electronics Engineers, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[223] Neus Sabater et al. “Dataset and Pipeline for Multi-view Light-Field Video”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 1743–1753. DOI: 10.1109/CVPRW.2017.221.

[224] Furkan E. Sahin and Rajiv Laroia. “Light L16 Computational Camera”. In: *Imaging and Applied Optics 2017 (3D, AIO, COSI, IS, MATH, pcAOP)*. Optica Publishing Group, 2017, JTu5A.20. DOI: 10.1364/3D.2017.JTu5A.20.

[225] Gabriel Salvador et al. “Efficient GPU-based implementation of the median filter based on a multi-pixel-per-thread framework”. In: Apr. 2018, pp. 121–124. DOI: 10.1109/SSIAI.2018.8470318.

[226] Jonatan Samuelsson. “The XVC Video Code”A Revolutionary Software-Defined Video Compression Format”. In: *SMPTE Motion Imaging Journal* 128.10 (2019), pp. 1–8. DOI: 10.5594/JMI.2019.2937737.

[227] Amadou Tidjani Sanda Mahama, Augustin S Dossa, and Pierre Gouton. “Choice of distance metrics for RGB color image analysis”. In: *Electronic Imaging* 2016 (Feb. 2016), pp. 1–4. DOI: 10.2352/ISSN.2470-1173.2016.20.COLOR-349.

[228] João M. Santos et al. “Lossless coding of light field images based on minimum-rate predictors”. In: *Journal of Visual Communication and Image Representation* 54 (2018), pp. 21–30. ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2018.03.003.

[229] Vadim V Sanzharov, Vladimir A Frolov, and Vladimir A Galaktionov. “Survey of nvidia rtx technology”. In: *Programming and Computer Software* 46 (2020), pp. 297–304. DOI: 10.1134/S0361768820030068.

[230] Daniel Scharstein, Richard Szeliski, and Ramin Zabih. “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”. In: *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*. 2001, pp. 131–140. DOI: 10.1109/SMBV.2001.988771.

[231] David C. Schedl and Oliver Bimber. “Optimized Sampling for View Interpolation in Light Fields with Overlapping Patches”. In: *EG 2018 - Short Papers*. Ed. by Olga Diamanti and Amir Vaxman. The Eurographics Association, 2018. DOI: 10.2312/egs.20181034.

[232] Michael Schmeing and Xiaoyi Jiang. “Depth image based rendering”. In: *Pattern Recognition, Machine Intelligence and Biometrics* (2011), pp. 279–310. DOI: 10.1007/978-3-642-22407-2_12.

[233] René Schuster et al. “Combining stereo disparity and optical flow for basic scene flow”. In: *Commercial Vehicle Technology 2018*. Springer, 2018, pp. 90–101. ISBN: 978-3-658-21300-8. DOI: 10.1007/978-3-658-21300-8_8.

[234] Heiko Schwarz, Thomas Schierl, and Detlev Marpe. “Block Structures and Parallelism Features in HEVC”. In: June 2014, pp. 49–90. ISBN: 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4_3.

[235] Gabriele Scrofani et al. “FIMic: design for ultimate 3D-integral microscopy of in-vivo biological samples”. In: *Biomed. Opt. Express* 9.1 (Jan. 2018), pp. 335–346. DOI: 10.1364/BOE.9.000335.

[236] Pieter Seuntiens, Lydia Meesters, and Wijnand Ijsselsteijn. “Perceived Quality of Compressed Stereoscopic Images: Effects of Symmetric and Asymmetric JPEG Coding and Camera Separation”. In: *ACM Trans. Appl. Percept.* 3.2 (Apr. 2006), 95–109. ISSN: 1544-3558. DOI: 10.1145/1141897.1141899.

[237] Sumit Shekhar et al. “Light-Field Intrinsic Dataset”. In: *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*. 2018, p. 120.

[238] Jinglei Shi, Xiaoran Jiang, and Christine Guillemot. “A framework for learning depth from a flexible subset of dense and sparse light field views”. In: *IEEE Transactions on Image Processing* 28.12 (2019), pp. 5867–5880. DOI: 10.1109/TIP.2019.2923323.

[239] Lixin Shi et al. “Light Field Reconstruction Using Sparsity in the Continuous Fourier Domain”. In: *ACM Trans. Graph.* 34.1 (Dec. 2015). ISSN: 0730-0301. DOI: 10.1145/2682631.

[240] Loren Shih. “Autofocus survey: a comparison of algorithms”. In: *Digital Photography III*. Ed. by Russel A. Martin, Jeffrey M. DiCarlo, and Nitin Sampat. Vol. 6502. International Society for Optics and Photonics. SPIE, 2007, 65020B. DOI: 10.1117/12.705386.

[241] Mohana Singh and Renu M. Rameshan. “Learning-Based Practical Light Field Image Compression Using A Disparity-Aware Model”. In: *2021 Picture Coding Symposium (PCS)*. 2021, pp. 1–5. DOI: 10.1109/PCS50896.2021.9477448.

[242] Sonali et al. “An approach for de-noising and contrast enhancement of retinal fundus image using CLAHE”. In: *Optics & Laser Technology* 110 (2019). Special Issue: Optical Imaging for Extreme Environment, pp. 87–98. ISSN: 0030-3992. DOI: 10.1016/j.optlastec.2018.06.061.

[243] Filippo Speranza et al. “Effect of disparity and motion on visual comfort of stereoscopic images”. In: *Stereoscopic displays and virtual reality systems XIII*. Vol. 6055. SPIE. 2006, pp. 94–103. ISBN: 0819460958.

[244] George Sperling and Barbara Anne Dosher. “Depth from motion”. In: *Early vision and beyond* (1994), pp. 133–142.

[245] K Srinivas Rao and AV Paramkusam. “Block Matching Algorithms for the Estimation of Motion in Image Sequences: Analysis”. In: *Pattern Recognition and Image Analysis* 32.1 (2022), pp. 33–44. DOI: 10.1016/j.procs.2021.09.070.

[246] Cecie Starr, Christine Evers, and Lisa Starr. *Biology: concepts and applications*. Cengage Learning, 2014. ISBN: 1285974654.

[247] K Sugita et al. “Focus Measurement on Programmable Graphics Hardware for All in-Focus Rendering from Light Fields”. In: *Virtual Reality Conference, IEEE*. Los Alamitos, CA, USA: IEEE Computer Society, Mar. 2004, p. 255. DOI: 10.1109/VR.2004.1310096.

[248] Norihiro Sugita et al. “Effect of viewing a three-dimensional movie with vertical parallax”. In: *Displays* 58 (2019). Special Issue: Visually Induced Motion Sensations, pp. 20–26. ISSN: 0141-9382. DOI: 10.1016/j.displa.2018.10.007.

[249] Mohammed Suhail et al. “Generalizable Patch-Based Neural Rendering”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 156–174. ISBN: 978-3-031-19824-3.

[250] Mohammed Suhail et al. “Light Field Neural Rendering”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8259–8269. DOI: 10.1109/CVPR52688.2022.00809.

[251] Gary J. Sullivan et al. “Overview of the High Efficiency Video Coding (HEVC) Standard”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), pp. 1649–1668. DOI: 10.1109/TCSVT.2012.2221191.

[252] Deqing Sun, Stefan Roth, and Michael J. Black. “Secrets of optical flow estimation and their principles”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2432–2439. DOI: 10.1109/CVPR.2010.5539939.

[253] Michael T. Swanston and Walter C. Gogel. “Perceived size and motion in depth from optical expansion”. In: *Perception & Psychophysics* 39.5 (Sept. 1986), pp. 309–326. ISSN: 1532-5962. DOI: 10.3758/BF03202998.

[254] Keita Takahashi, Akira Kubota Takeshi Naemura, and Takeshi Naemura. “All in-focus view synthesis from under-sampled light fields”. In: *Proc. Int. Conf. Artificial Reality and Telexistence (ICAT 2003), Dec.* (2003), pp. 249–256.

[255] Yasuhiro Takaki and Nichiyo Nago. “Multi-projection of lenticular displays to construct a 256-view super multi-view display”. In: *Optics express* 18.9 (2010), pp. 8824–8835. DOI: 10.1364/OE.18.008824.

[256] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: A survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), pp. 1–11. DOI: 10.1186/s41074-017-0027-2.

[257] Michael W. Tao et al. “Depth from Combining Defocus and Correspondence Using Light-Field Cameras”. In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 673–680. DOI: 10.1109/ICCV.2013.89.

[258] Kasim Terzić and Miles Hansard. “Methods for reducing visual discomfort in stereoscopic 3D: A review”. In: *Signal Processing: Image Communication* 47 (2016), pp. 402–416. ISSN: 0923-5965. DOI: 0.1016/j.image.2016.08.002.

[259] Shishun Tian et al. “NIQSV+: A No-Reference Synthesized View Quality Assessment Metric”. In: *IEEE Transactions on Image Processing* 27.4 (2018), pp. 1652–1664. DOI: 10.1109/TIP.2017.2781420.

[260] Fabrizio Tiburzi and Jesús Bescós. “Camera Motion Analysis in On-line MPEG Sequences”. In: *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS ’07)*. Institute of Electrical and Electronics Engineers, 2007, pp. 42–42. DOI: 10.1109/WIAMIS.2007.27.

[261] Severin Todt, Christof Rezk Salama, and Andreas Kolb. “Light Field Rendering for Games”. In: *Theory and Practice of Computer Graphics*. Ed. by Ik Soo Lim and Wen Tang. The Eurographics Association, 2008. ISBN: 978-3-905673-67-8. DOI: 10.2312/LocalChapterEvents/TPCG/TPCG08/027-033.

[262] Severin Todt et al. “Fast (spherical) light field rendering with per-pixel depth”. In: *Technical report* (2007).

[263] Marc Comino Trinidad et al. “Multi-View Image Fusion”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 4100–4109. DOI: 10.1109/ICCV.2019.00420.

[264] Jonas Trottnow et al. “The Potential of Light Fields in Media Productions”. In: *SIGGRAPH Asia 2019 Technical Briefs*. SA ’19. Brisbane, QLD, Australia: Association for Computing Machinery, 2019, 71–74. ISBN: 9781450369459. DOI: 10.1145/3355088.3365158.

[265] Tom Tullis and Bill Albert. “Chapter 1 - Introduction”. In: *Measuring the User Experience (Second Edition)*. Ed. by Tom Tullis and Bill Albert. Second Edition. Interactive Technologies. Boston: Morgan Kaufmann, 2013. ISBN: 978-0-12-415781-1. DOI: 10.1016/B978-0-12-415781-1.00001-7.

[266] Suren Vagharshakyan, Robert Bregovic, and Atanas Gotchev. “Light Field Reconstruction Using Shearlet Transform”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.1 (Jan. 2018), pp. 133–147. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2017.2653101.

[267] Vaibhav Vaish and Andrew Adams. “The (new) stanford light field archive”. In: *Computer Graphics Laboratory, Stanford University* 6.7 (2008).

[268] Irene Viola, Martin Rerabek, and Touradj Ebrahimi. “Comparison and Evaluation of Light Field Image Coding Approaches”. In: *IEEE Journal of Selected Topics in Signal Processing* 11.7 (2017), pp. 1092–1106. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2017.2740167.

[269] Christian Vogelgsang and Günther Greiner. “Adaptive lumigraph rendering with depth maps”. In: *Technical Report 3, IMMD 9, Universitaet Erlangen-Nuernberg* (2000).

[270] Gregory K. Wallace. “The JPEG still picture compression standard”. In: *IEEE Transactions on Consumer Electronics* 38.1 (1992), pp. xviii–xxxiv. DOI: 10.1109/30.125072.

[271] Gengkun Wang et al. “Light Field Multi-View Video Coding With Two-Directional Parallel Inter-View Prediction”. In: *IEEE Transactions on Image Processing* 25.11 (2016), pp. 5104–5117. DOI: 10.1109/TIP.2016.2603602.

[272] Huamin Wang, Mingxuan Sun, and Ruigang Yang. “Space-Time Light Field Rendering”. In: *IEEE transactions on visualization and computer graphics* 13 (Aug. 2007), pp. 697–710. DOI: 10.1109/TVCG.2007.1019.

[273] Lijun Wang et al. “DeepLens: shallow depth of field from a single image”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275013. URL: https://doi.org/10.1145/3272127.3275013.

[274] Qianqian Wang et al. “IBRNet: Learning Multi-View Image-Based Rendering”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2021, pp. 4688–4697. DOI: 10.1109/CVPR46437.2021.00466.

[275] Qin Wang, Hui Cheng, and Lei Zhang. “Assessment of Individual Differences for Aftereffect of Viewing Autostereoscopic Display”. In: *Available at SSRN 4339603* (2023). DOI: 10.1002/jsid.1265.

[276] Shuhui Wang et al. “Shot classification for action movies based on motion characteristics”. In: *2008 15th IEEE International Conference on Image Processing*. 2008, pp. 2508–2511. DOI: 10.1109/ICIP.2008.4712303.

[277] Ting-Chun Wang et al. “Light Field Video Capture Using a Learning-Based Hybrid Imaging System”. In: *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073614.

[278] Zirui Wang et al. “NeRF−−: Neural Radiance Fields Without Known Camera Parameters”. In: *arXiv preprint arXiv:2102.07064* (2021). DOI: 10.48550/arXiv.2102.07064.

[279] Sven Wanner and Bastian Goldluecke. “Variational Light Field Analysis for Disparity Estimation and Super-Resolution”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.3 (2014), pp. 606–619. DOI: 10.1109/TPAMI.2013.147.

[280] Michael Waschbüsch, Stephan Würmlin, and Markus Gross. “3d video billboard clouds”. In: *Computer Graphics Forum*. Vol. 26. 3. Wiley Online Library. 2007, pp. 561–569. DOI: doi.org/10.1111/j.1467-8659.2007.01079.x.

[281] Barry Payne Welford. “Note on a Method for Calculating Corrected Sums of Squares and Products”. In: *Technometrics* 4.3 (1962), pp. 419–420. DOI: 10.1080/00401706.1962.10490022.

[282] Ying Weng and Jianmin Jiang. “Fast Camera Motion Estimation in MPEG Compressed Domain”. In: *Consumer Electronics, IEEE Transactions on* 57 (Sept. 2011), pp. 1329–1335. DOI: 10.1109/TCE.2011.6018891.

[283] Bennett Wilburn et al. “High Performance Imaging Using Large Camera Arrays”. In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH ’05. Los Angeles, California: Association for Computing Machinery, 2005, 765–776. ISBN: 9781450378253. DOI: 10.1145/1186822.1073259.

[284] Lee Williams. “Imaging in a new dimension: Holographic and 3D visualisation companies are finding new ways to take the quality of holograms to the next level”. In: *Engineering & Technology* 14.11/12 (2019), pp. 64–67. DOI: 10.1049/et.2019.1206.

[285] Andrew Woods, Tom Docherty, and Rolf Koch. “Image Distortions in Stereoscopic Video Systems”. In: *Proc SPIE* 1915 (Nov. 2002). DOI: 10.1117/12.157041.

[286] Zhaolin Xiao et al. “Axial refocusing precision model with light fields”. In: *Signal Processing: Image Communication* 106 (2022), p. 116721. ISSN: 0923-5965. DOI: 10.1016/j.image.2022.116721.

[287] Qiwei Xing, Chunyi Chen, and Zhihua Li. “Progressive path tracing with bilateral-filtering-based denoising”. In: *Multimedia Tools and Applications* 80 (2021), pp. 1529–1544. DOI: 10.1007/s11042-020-09650-7.

[288] Wei Xiong and John Chung-Mong Lee. “Efficient scene change detection and camera motion annotation for video classification”. In: *Computer vision and image understanding* 71.2 (1998), pp. 166–181. DOI: 10.1006/cviu.1998.0711.

[289] Gang Xu and Zhengyou Zhang. *Epipolar geometry in stereo, motion and object recognition: a unified approach*. Vol. 6. Springer Science & Business Media, 2013. ISBN: 978-90-481-4743-4. DOI: 10.1007/978-94-015-8668-9.

[290] Zhou Xue. *Sampling models in light fields*. Tech. rep. EPFL, 2016. DOI: 10.5075/epfl-thesis-7003.

[291] Hirokazu Yamanoue, Makoto Okui, and Ichiro Yuyama. "A study on the relationship between shooting conditions and cardboard effect of stereoscopic images". In: *IEEE Transactions on Circuits and Systems for Video Technology* 10.3 (2000), pp. 411–416. DOI: 10.1109/76.836285.

[292] Lin Yang et al. "See in 3D: state of the art of 3D display technologies". In: *Multimedia Tools and Applications* 75.24 (2016), pp. 17121–17155. DOI: 10.1007/s11042-015-2981-y.

[293] Ren Yang et al. "Learning for Video Compression with Hierarchical Quality and Recurrent Enhancement". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. DOI: 10.1109/CVPR42600.2020.00666.

[294] Ola Younis et al. "Real-time Detection of Wearable Camera Motion Using Optical Flow". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, pp. 1–6. DOI: 10.1109/CEC.2018.8477783.

[295] Cha Zhang and Tsuhan Chen. "The Self-Reconfigurable Camera Array". In: *Light Field Sampling*. Cham: Springer International Publishing, 2006, pp. 67–86. ISBN: 978-3-031-02241-8. DOI: 10.1007/978-3-031-02241-8_6.

[296] Weixia Zhang et al. "Blind Image Quality Assessment via Vision-Language Correspondence: A Multitask Learning Perspective". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2023. DOI: 10.48550/arXiv.2303.14968.

[297] Zhao Zhang et al. "Gradient-Induced Co-Saliency Detection". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 455–472. ISBN: 978-3-030-58610-2.

[298] Zhengyou Zhang. "Microsoft kinect sensor and its effect". In: *IEEE multimedia* 19.2 (2012), pp. 4–10. DOI: 10.1109/MMUL.2012.24.

[299] Jinbo Zhao et al. "Light Field Image Compression via CNN-Based EPI Super-Resolution and Decoder-Side Quality Enhancement". In: *IEEE Access* 7 (2019), pp. 135982–135998. DOI: 10.1109/ACCESS.2019.2930644.

[300] Yan Zhou et al. "3D reconstruction based on light field information". In: *2015 IEEE International Conference on Information and Automation*. 2015, pp. 976–981. DOI: 10.1109/ICInfA.2015.7279428.

[301] Matthias Zwicker et al. "Antialiasing for Automultiscopic 3D Displays". In: *Symposium on Rendering*. Ed. by Tomas Akenine-Moeller and Wolfgang Heidrich. The Eurographics Association, 2006. ISBN: 3-905673-35-5. DOI: 10.2312/EGWR/EGSR06/073-082.

[302] Matthias Zwicker et al. "Resampling, Antialiasing, and Compression in Multiview 3-D Displays". In: *IEEE Signal Processing Magazine* 24.6 (2007), pp. 88–96. DOI: 10.1109/MSP.2007.905708.