UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Depth estimation for light field images and light field image synthesis

by Rishabh Sharma

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Dr. Eva Cheng and Dr. Stuart Perry

University of Technology Sydney
Faculty of Engineering and Information Technology

April 2024

Certificate of Original Authorship

I, Rishabh Sharma, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the FEIT.School of Professional Practice and Leadership at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signed: Rishabh Sharma

Date: 16th *October*, 2022

Depth estimation for light field images and light field image synthesis

by

Rishabh Sharma

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

Light field imaging technology allows capturing the pixel intensity and the direction of the incident light with a single capture. Capturing this additional dimensionality allows for generating images at different focal lengths, extending the depth of field, and estimating the scene depth from a single capture. Depth estimation for light field images is fundamental for various light field applications such as light field image compression techniques, reconstructing views from a sparse set of perspective views, and 3D reconstruction. Although depth estimation for light field images allows researchers to explore techniques such as depth from correspondence and defocus or using epipolar plane images from single image capture, the same challenges apply as for depth estimation techniques using a single image or stereo image pair, such as occlusions, textureless and overexposed regions.

This thesis presents an algorithm to improve depth map estimation for light field images using depth from defocus. Previous depth map estimation approaches for light field images do not capture sharp transitions around object boundaries due to occlusions, making many current approaches unreliable at depth discontinuities. This is especially the case for light field images because the pixels do not exhibit photo-consistency in the presence of occlusions. In this work, a small patch size of pixels is used in each focal stack image for comparing defocus cues, allowing the algorithm to generate sharper depth boundaries.

Then, in contrast to existing approaches that use defocus cues for depth estimation, frequency domain analysis image similarity checking then generates the depth map. Processing in the frequency domain reduces the individual pixel errors that occur while directly comparing RGB images, making the algorithm more resilient to noise.

In addition to evaluating this depth map estimation approach on both a synthetic image dataset and real-world images in the JPEG dataset, the depth map estimation is applied to light field synthesis to evaluate real-world application of the presented approach. The application of light field synthesis was chosen because the capture of light field images is non-trivial, due to the need for multiple viewpoints. Two well-known ways to capture a light field are: to build a camera array with multiple cameras to capture the image from different viewpoints; or, to move a camera through the scene to capture the different viewpoints. This second approach uses a plenoptic camera that places a microlens array in front of the conventional 2D camera sensor. However, both of these techniques have drawbacks: the multicamera array or moving camera exhibit high aliasing in blurred regions while generating refocused images and are expensive in terms of the hardware and time taken to capture, whereas the plenoptic camera has a trade-off between angular and spatial resolution. Light field reconstruction and synthesis algorithms are thus essential for improving the lower spatial resolution of hand-held plenoptic cameras.

The depth estimation techniques explored for light field images form the building blocks for solving this inverse problem of light field image reconstruction and synthesis. The ability to convert 2D RGB images to 4D light field images will change how we perceive traditional photography. Previous light field synthesis algorithms produce blurred regions around depth discontinuities, especially for stereo-based algorithms, where no information can fill the occluded areas in the light field image. This thesis presents a light field synthesis algorithm that uses the focal stack images and the all-in-focus image to synthesize a 9 x 9 sub-aperture view light field image. Using the presented approach of estimating a depth map using depth from defocus, this depth map and the all-in-focus image synthesize the sub-aperture views and their corresponding depth maps by mimicking the apparent shifting of the central image according to the depth values. Occluded regions are handled in the synthesized sub-aperture views by filling them with the information recovered from the focal stack images. Experimental results demonstrate that the presented algorithm outperforms state-of-the-art depth estimation techniques for light field images, particularly in the case of noisy images. Results also show that if the depth levels in the image are known, a high-accuracy light field image can be synthesized with just five focal stack images.

Acknowledgements

I want to start by thanking my supervisor, Dr. Eva Cheng; without her support and encouragement, I wouldn't have been able to embark on this life-changing experience. I am eternally grateful to her for this. I am sure I am just one of the many people whose lives she has helped to change for the better. I would like to express my sincere gratitude to my supervisors, Dr. Eva Cheng and Dr.Stuart Perry, for their patience, kindness and continuous support. Their wisdom and knowledge have made me a better researcher. I will miss their guidance and encouragement and also Stuart's wit. I could not have imagined having better mentors for my Ph.D. study.

My Parents, Mahinder and Rajrani, have been the two pillars of my life. I would have done some pretty good things in my previous life to be blessed with them as my parents. I believe that their selfless attitude to helping others is why I have always met people in my life who have helped me reach where I am today. Since I can remember, they have trusted me with my judgment and given me the freedom to learn from my own mistakes. I hope to one day become as humble, kind, and selfless as them. I would like to mention my Grandparents, Nikkaram and Shankari Devi, for their progressive thinking, sacrifice, and hard work ensuring that the next generation could have comfortable lives. I thank my aunties, Kuldeep, Pradeep and Radhika, for their love and support. A special mention to my uncle, Sudhir, who instilled my passion for technology.

My elder sisters, Dimple and Kiran, have always been there for me through my ups and downs. I could talk to them about anything and everything, and they were always able to show me the right path. Ruchi and Neeraj, for their support. Nitish helped me bounce off ideas about my research and contemplate the more critical questions in life. Talking to him was always refreshing, and he motivated and inspired me to keep improving my research.

My family here in Australia, Ravi, Gerry, Mayur, Prateek, Tanmay, Jibu, Giridhar, and Saurabh, have always been there for me, and I feel lucky to have met them, and that they have let me be part of their lives. My friend Sanam for showing me how to stay positive through life, being there for me whenever I have found myself in doubt, and always believing in me.

I had moved from Melbourne to Sydney to pursue my research, and with all the uncertainties of being in a new place, my roommates from Chippendale, Nithila, Urjit, Martyna and Priyanshi, helped me find a place to call home. Nithila, an architecture graduate, showed me all the parts of Sydney from an art and cultural perspective, helping me appreciate Sydney even more.

Covid was one of the most challenging times in the world, and frankly, with the lockdowns, our minds had become our worst enemy for many of us. I feel privileged to have met Manju, Saurabh, Neeraj, Dhananjay and Srirag during these times, and I couldn't have hoped to find better people than them to help me during these difficult times.

I would also like to thank Nina, Manoj and Ayush for providing the support I needed during the last few months of my research, and I am forever grateful to them for this.

I don't think my words have done justice to express how much all these people have helped me. None of this would have been possible if these people hadn't been a part of my journey.

Contents

List of Figures

List of Tables

List of Publications

1. **Noise-Resilient Depth Estimation for Light Field Images Using Focal Stack and FFT Analysis [14]**
2. **Light field view synthesis using the focal stack and all-in-focus image (Accepted) [15]**

Chapter 1

Introduction

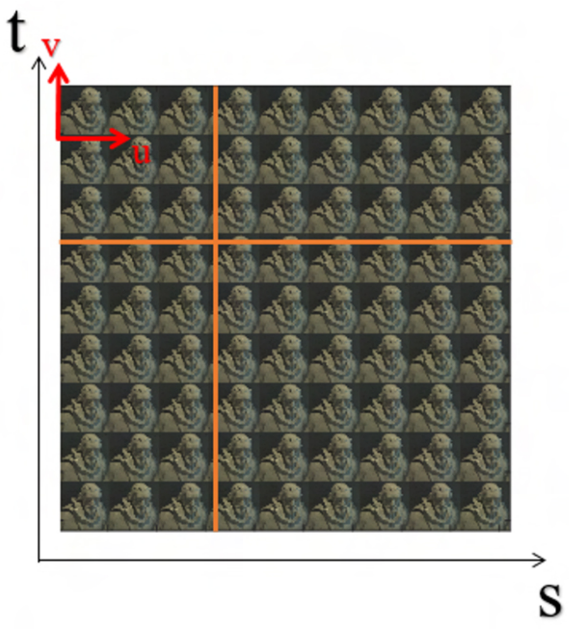
Light field imaging technology captures rich visual information by representing light distribution in free space in contrast to conventional photography, which only records the sum of the intensity of light rays striking each point in the image [16]. Effectively this means that the light field images capture not only the pixel intensity but also the direction of the incident light with a single capture. The additional data dimensions captured enables ray-tracing techniques to generate images at different focal lengths and extended field depth, allowing image manipulation in a more flexible way [17].

In recent years, the introduction of hand-held light field cameras, also known as the plenoptic camera, has made it possible to capture light field images with a single capture. One method of creating a plenoptic cameras is by placing a microlens array in front of a conventional 2D camera sensor. Before the introduction of hand-held light field cameras, light field images were captured by building a camera array with multiple cameras to capture the image from different viewpoints; or moving a camera through the scene to capture different viewpoints. Though plenoptic cameras solve the problem of capturing a light field image in a superior way compared to a camera array in terms of hardware expense and time taken to capture, the plenoptic camera has a trade-off between angular and spatial resolution [18].

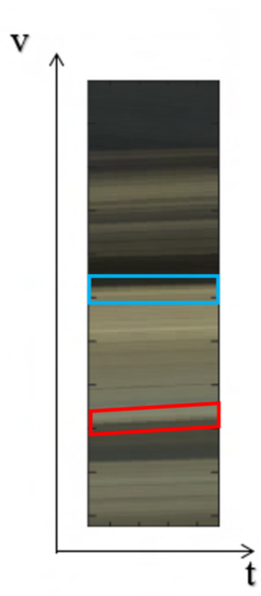
The higher dimensional data captured in the light field image helps improve the performance of the computer vision problems encountered for 2D images, such as depth estimation, post-capture refocusing, and image segmentation [16]. Depth maps from light field images are essential for light field image compression techniques, reconstructing views from a sparse set of perspective views, increasing the number of perspective views, and 3D reconstruction. More generally, depth estimation for 2D planar images is essential for applications such as changing depth of focus, simulating subsurface scattering and shadow mapping. Depth estimation is also crucial in computer vision applications such as robot vision, self-driving cars, surveillance and human-computer interactions. It plays an important role in semantic segmentation algorithms [19, 20]. A wide range of stereo-matching techniques have been proposed and implemented for depth estimation, and Scharstein and Szeliski present an in-depth analysis of these techniques [21].

Compared to techniques using a single image or stereo image pair for depth estimation, light field images enable researchers to explore techniques such as depth from correspondence and defocus from a single image. However, the challenges in depth map estimation from planar images still apply, and are compounded by the challenges of estimating depth from light field images, including the presence of occlusion, textureless regions and over-exposed regions in images.

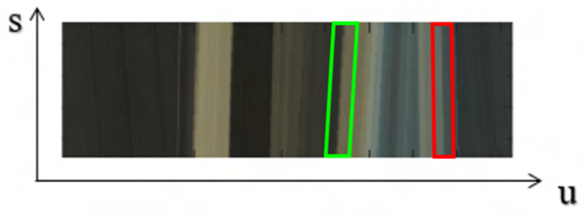
Existing light field depth estimation techniques include stereo-based matching, using epipolar images, depth from defocus and more recent techniques that use neural networks. Stereo matching for depth estimation suffers from ambiguity due to partial occlusions [22]. Since the stereo pair are images, the information lost by the occlusions cannot be recovered but can only be approximated using a smoothness term to fill the gaps. In contrast, epipolar images are formed by stacking together light field sub-aperture images in both the horizontal and vertical direction as shown in Figure 1.1(B) and Figure 1.1(C), respectively [16]. A slice through this 4D block reveals the depth of the pixels, as the slope of the line reflects the depth information as shown by the red, green and blue parallelograms in Figure 1.1. The pixels in the central view that do not move with changing sub-aperture views have a zero slope and are seen as a straight line as shown by the blue parallelograms in Figure 1.1(B). The pixels that are closer to the camera incline to the right as shown by the green parallelograms in Figure 1.1(C), and the pixels that are further away from



(A)



(B)



(C)

Figure 1.1: (A) A sub-aperture image view; (B) the EPI for the vertical line represented in (A); (C) the EPI for the horizontal line represented in (A).

the camera incline to the left as shown by the red parallelograms in Figure 1.1. However, basic line fitting techniques to estimate depth do not give robust results and the reconstructions are generally noisy [23]. Schechner and Kiryati [24] have extensively studied depth from defocus and correspondence techniques and have compared the advantages and disadvantages of both cues. Finally, Convolutional Neural Networks (CNN) are a known approach in imaging applications such as object recognition, human–computer interaction, image segmentation and stereo depth estimation [25]. However, the lack of training data in existing light field images datasets makes it hard to train and test the network for light field images [26].

This thesis explores depth estimation from light field images, using the application of light field image synthesis to evaluate the proposed approaches. This is because the techniques explored for depth estimation of light field images form the building blocks for solving the inverse problem of light field image reconstruction and synthesis. Light field reconstruction and synthesis algorithms can also address the problem of lower spatial resolutions for hand-held plenoptic cameras used for light field image capture, where the ability to convert 2D RGB images to 4D light field images will change how we perceive traditional photography.

Many algorithms that propose light field reconstruction techniques use a sparse set of light field views to reconstruct novel views [27–30]. However, the input data for these algorithms is a set of sub-aperture images, which is not easy to capture because the camera needs to be moved to capture the sub-aperture views using a 2D camera, and this is time-consuming and introduces issues of alignment. In contrast, for capturing a focal stack and all-in-focus images, we only need to change the focus and aperture of the camera without physically moving the camera. Even though these algorithms can be used for light field synthesis, due to the complexity of capturing these sub-aperture images, using these algorithms for light field synthesis in real-world scenarios would be difficult. Whereas, they will be ideal for either increasing the spatial and angular resolution of light field images or for light field image compression [29].

On the other hand, approaches for light field synthesis can be classified into two main categories, non-learning-based and learning-based approaches. Non-learning-based light field synthesis algorithms use a deterministic approach, where the same rules are used to synthesize the view for every image. These synthesis algorithms can be further divided into two categories based on the input data used: focal stack images, or stereo image pairs. Algorithms that used stereo image pairs either use micro-baseline image pairs or an image pair with a large baseline. Zhang *et al.* [9] propose a micro-baseline image pair-based view synthesis algorithm. Since the disparity between the stereo pair is small, the images can be captured by vibrating a static camera. Chao *et al.* [11] on the other hand, use a large baseline stereo pair. As the algorithm uses a large baseline, the horizontal views are synthesized by interpolating the stereo pair. In contrast, the main advantage of using focal stack images for light field synthesis is that the focal stack images can provide information to fill the gaps created near occluded regions in the synthesized views [28, 31, 32].

Learning-based light field synthesis algorithms use a probabilistic approach, where the training input images are used to fit a model that can map the output. The two main drawbacks of learning-based light field synthesis approaches are: first, a large amount of training data is required to train the network adequately; second, that the algorithm's accuracy directly relates to the training data quality. Some learning-based algorithms [10, 11, 33] synthesize the entire 9 x 9 sub-aperture light field images using two, four, and one input image, respectively. Although these algorithms use fewer images as input, the Convolutional Neural Network (CNN) must be trained on a significant amount of training data to ensure high accuracy.

1.1 Thesis statement

This thesis focuses on depth estimation algorithms for light field images using focal stack images. We focus on two main issues encountered by existing depth estimation algorithms for light field images: depth inaccuracies around occlusions, and depth estimation accuracy for noisy images. As depth estimation techniques for light field images are essential for analyzing the inverse problem of light field image reconstruction and synthesis, we then extend our depth estimation algorithm by synthesizing the light field image using the focal stack and estimated depth map. We specifically focus on solving the inverse problem for light field synthesis using the focal stack and all-in-focus image, to address issues existing synthesis algorithms face in filling the occluded regions in the synthesized views and improve the accuracy of the synthesized sub-aperture views near depth discontinuities.

1.1.1 Depth estimation for light field images

The research questions that we address in this thesis are:

- How can depth estimation techniques improve around the complex imaging conditions of occluded boundaries?
- How can depth estimation techniques improve in noise resilience?

Depth estimation algorithms typically fail to estimate the depth accurately around depth discontinuities due to occlusions. To improve the depth estimation for light field images, we propose a simple but effective depth-from-defocus algorithm that uses the focal stack and all-in-focus image. We chose a focal stack-based depth estimation approach as it offers distinct advantages over both sub-aperture and Epipolar Plane Image (EPI) approaches in the context of light field imaging. Like those employed in multi-view stereo methods, focal stack methodologies leverage a crucial advantage: occlusion tends to manifest in a singular direction, perpendicular to the focal plane within the stack. This characteristic enables comparisons among occlusion-free sections of the stack, enhancing depth estimation accuracy by utilizing unobstructed views for comparison [34].

On the contrary, depth estimation using sub-aperture views suffers from drawbacks. Stereo-based algorithms encounter ambiguity issues, particularly when handling noisy images. Moreover, the limited baseline in real camera light field images complicates occlusion resolution, posing a challenge for these algorithms to effectively tackle occlusion problems [7].

Similarly, EPI-based techniques are insensitive to occlusion, noise, angular resolution, spatial aliasing, and broad depth ranges. However, in the case of noisy images, EPI-based approaches often generate outliers, significantly compromising the fidelity of their depth maps [14]. Additionally, integrating estimates from horizontal and vertical EPIs remains a persistent challenge across all EPI-based methodologies, posing a fundamental problem in these approaches.

In order to address the issues around the complex imaging conditions of occluded boundaries, we look at how the focal stack images are generated for a light field image. We generate the focal stack from light field images using the median value of the shifted sub-aperture images instead of taking the average value. By doing this, we ensure that the images focusing on the foreground do not have the defocus blur around the depth discontinuities, improving our algorithm's accuracy near depth boundaries.

The other challenge with existing depth estimation algorithms is that their accuracy drastically drops in the presence of noise; few depth map estimation approaches currently focus on emphasis noise resilience. To improve the noise resilience of depth estimation using a focal stack, we use the frequency domain analysis instead of the RGB domain to compare the focal stack image regions with the all-in-focus image, ensuring our algorithm's noise resilience.

Unlike other depth estimation approaches for light field images that cannot be explicitly extended for 2D images, since our approach uses depth-from-defocus, we can easily extend our approach to estimate the depth map for the focal stack captured by a 2D camera. To evaluate the parameters that need to be considered to capture the focal stack for light field synthesis, we capture the focal stack with a 2D camera using different f-stop and step sizes to evaluate the best parameters for accurate depth map estimation. The difference in the f-stop used for capturing the focal stack and all-in-focus images causes the lighting conditions to change slightly, which makes using only the frequency domain for comparison unreliable, especially for smooth texture regions in the image. We address this issue by combining the frequency spectrum of the image regions with the sharpness and gradient of the region, and use the combined confidence score to estimate the depth map.

1.1.2 Light field synthesis

The research questions that we address in this thesis are:

- How can light field image synthesis approaches improve the synthesis accuracy, especially for occluded regions?
- Which parameters need to be considered to capture the focal stack for light field synthesis?

Light field synthesis approaches either use focal stack images or stereo image pairs as input. The main issue with using a stereo image pair as input data for light field synthesis is that stereo-pair only captures the horizontal parallax. So these algorithms are able to synthesize the horizontal views within the stereo-pair baseline accurately, but as no information is available in the vertical direction, their accuracy reduces for vertical views. To improve the synthesis accuracy for occluded regions, we use the defocus blur in focal stack images. The defocus blur in focal stack images reveals parts of the background that are not visible in the all-in-focus image in both the horizontal and vertical directions. Our algorithm synthesizes the light field image using the focal stack and all-in-focus image. In our approach, the focal stack and all-in-focus image are first used to estimate the depth map. The depth information is then used to mimic the apparent shifting of the all-in-focus image according to the depth values to synthesize the sub-aperture views. The gaps generated in the synthesized views due to occlusions are filled with the information extracted from the defocused regions in the focal stack, ensuring the synthesized sub-aperture accuracy for both horizontal and vertical views.

1.2 Contributions

1.2.1 Depth estimation for light field images

1. To reduce the dependence of depth accuracy on RGB values of individual pixels compared in the image patches, our proposed depth estimation approach uses frequency domain analysis to estimate the depth map for light field images.

2. The key contribution of our approach is noise resilience in depth estimation: our analysis confirms the hypothesis that comparing focal stack image patches in the frequency domain improves depth map accuracy, especially in the presence of noise.

We propose a depth estimation algorithm that uses depth-from-defocus to estimate the depth maps. Specifically, we divide the focal stack and all-in-focus image into smaller patches of size 4 x 4 pixels and compare the FFTs of these patches to find the closest match to the all-in-focus image patch. We show that using FFTs of the patch instead of the RGB patches makes the algorithm resilient to noise. For generating the focal stack from light field images, we shift the sub-aperture images in the frequency domain and use the median value of the pixels. We show that using the frequency domain ensures sub-pixel accuracy for the focal stack when generating them even at small focal distances, increasing the depth precision for our depth maps. The median values reduce the defocus blur near the depth discontinuities and ensure accurate depth estimation for occluded regions.

We also extend our depth estimation for the focal stack captured by a 2D camera, focusing on light field synthesis using focal stacks. We show that capturing the focal stack with an f-stop of f/1.8 and a step size of 100 covers the scene with sufficient depth levels to synthesize a light field image using the depth map and all-in-focus image. When capturing a focal stack, images are taken at varying focal distances. The step size specifically denotes the magnitude of change in the camera's focus position between each consecutive shot in the stack. For instance, if the step size is 100, the camera incrementally adjusts its focus in units corresponding to the camera's focal distance scale, with each step representing a change of 100 units between maximum and minimum focus points, allowing the camera to capture 21 images in the focal stack.

In the test scenario described capturing a focal stack with a f-stop of f/1.8 and a step size of 100 the choice of the step size (100) determines the granularity or spacing between the different focal planes captured. A smaller step size results in a finer sampling of depth information but might require more images to cover the desired depth range, increasing computational complexity and storage requirements. Conversely, a larger step size covers a broader range with fewer images but might potentially miss finer depth details.

In this case, the selection of a step size of 100 is considered sufficient to cover the scene with an appropriate number of depth levels. It indicates that the focal stack comprises images captured at intervals that span 100 units of focal distance between each consecutive shot. This step size is deemed adequate to generate a comprehensive representation of the scene's depth, enabling subsequent synthesis of a light field image using the acquired depth map and an all-in-focus image.

We also extend the work on the existing light field camera simulation model in Blender, to allow researchers to capture the focal stack of the same light field image with varying camera parameters like image resolution, f-stop and depth of field. We believe that our proposed toolkit will enable advancements in CNN and deep learning approaches that use focal stack images for light field synthesis.

1.2.2 Light field synthesis

1. To fill the occluded regions with the information recovered from the focal stack images, our light field synthesis approach synthesizes high-accuracy light field images with varying sizes of focal stacks as input.

2. Using the frequency domain to mimic the apparent movement of the regions at different depths in the sub-aperture view, our approach ensures sub-pixel accuracy for small depth values.

We propose a light field synthesis algorithm that uses the focal stack images and the all-in-focus image to synthesize a 9 x 9 sub-aperture view light field image. We use depth from defocus to estimate a depth map. Then we use the depth map and the all-in-focus image to synthesize the sub-aperture views. We show that the algorithm can synthesize high-accuracy light field images with a varying number of focal stack images. The defocus blur in the focal stack image reveals parts of the background that are not visible in the all-in-focus image, which we use to fill the gaps in occluded regions in the horizontal and vertical synthesized view. We also show that using the frequency domain to mimic the apparent movement of the regions at different depths in the sub-aperture view ensures sub-pixel accuracy for small depth values.

1.3 Thesis overview

In this thesis, Chapter 2 introduces the theory of light field imaging technology with Chapter 3 presenting the existing work on depth estimation for light field images and light field synthesis techniques. Chapter 4 presents our extension to the existing light field toolbox to capture focal stack images with varying parameters of the same scene, enabling the evaluation of depth map estimation and light field synthesis algorithms. Chapter 5 presents our depth estimation algorithm for light field images using focal stack images, and the experimental validation of the algorithm's noise resilience. Chapter 6 then extends on the work of Chapter 5 to develop a depth estimation algorithm for focal stacks captured with a 2D camera, rather than a light field image. Finally, Chapter 7 extends on the work from Chapters 5 through 7, to present our light field synthesis algorithm that utilizes our depth map estimation algorithm, with experimental validation of the improvements over existing techniques. Chapter 8 thus conclusions this thesis and suggests future directions for the work.

Chapter 2

Background

2.1 Fundamentals concepts in photography

Before delving into light field technology, presented here is background with several imaging concepts that are referenced in subsequent chapters.

2.1.1 Focal Length

Focal length is the distance between the lens and the image sensor when the subject is in focus. It determines the magnification and angle of view of the captured scene. It's often denoted in millimeters (mm). For example, shorter focal lengths (e.g., 24mm) encompass wider angles, while longer ones (e.g., 200mm) provide narrower fields of view.

2.1.2 Focal Point

The focal point is the specific point at which light rays converge after passing through the lens. It's where the image appears sharp and in focus. In a simplified thin lens model, it's the point where parallel light rays converge after passing through the lens.

2.1.3 Aperture (f-stop)

Aperture refers to the opening in the lens that controls the amount of light entering the camera. The f-stop represents the size of the aperture. Smaller f-stop numbers (e.g., f/2.8) denote larger apertures, allowing more light to enter the camera, whereas higher f-stop numbers (e.g., f/16) indicate smaller apertures, restricting light entering the camera.

2.1.4 Depth of Field

Depth of field is the range of distances within a scene that appear acceptably sharp in an image. It's influenced by aperture size, focal length, and subject distance. A larger aperture (lower f-stop) creates a shallower depth of field, whereas a smaller aperture (higher f-stop) results in a deeper depth of field.

2.1.5 Circle of Confusion

This is the measure of the largest acceptable spot that a point in the scene can be focused onto the camera sensor while still being perceived as a point. This is a factor in determining depth of field.

2.1.6 Intrinsic and Extrinsic Parameters

Intrinsic parameters refer to internal camera characteristics (e.g., focal length, optical center), while extrinsic parameters relate to the camera's position and orientation in the 3D world.

2.1.7 Baseline

Baseline refers to the distance between the optical centers of two cameras in a stereo vision system. It influences depth perception and accuracy in 3D reconstruction.

2.1.8 Frustum

In computer graphics and geometry, the frustum represents the portion of space in the shape of a pyramid with its top removed. It's commonly used to define the view volume for rendering in 3D graphics.

2.1.9 Focal Stacking

Focal stacking involves taking multiple images with varying focus distances and combining them to create a final image with extended depth of field.

2.2 Light field imaging technology

Conventional photography is only able to capture limited information from the light passing through the camera lens. In general, cameras record the sum of the intensities of light rays striking each point in the image and not the total amount of incident light traveling along different rays that contribute to the image [35]. In contrast, light field imaging technology captures rich visual information by representing the distribution of light in free space [16], which means that a light field image captures the pixel intensity and the direction of the incident light. The additional dimensions of data captured enables the generation of focal stacks, and extended depth of field using ray-tracing techniques. A focal stack is a set of images taken by varying the focus sequentially either towards or away from the camera for each image.

The model that describes the distribution of light is known as a plenoptic function. The plenoptic function describes light as a function of position, angle of incidence, wavelength and time [16]. The plenoptic function represents the measurement of light rays at every possible location in space *(x, y, z)* , at every angle *(θ, ϕ)*, for every wavelength γ and at every time t [36]. The plenoptic function is thus a 7D function represented as *L(x, y, z, θ, ϕ,γ, t)*. Since it is not easy to capture and handle such high dimensional data in practice, we can simplify the function by reducing its dimensionality. By assuming the captured image to be time-invariant, we can reduce the plenoptic function to a 5D

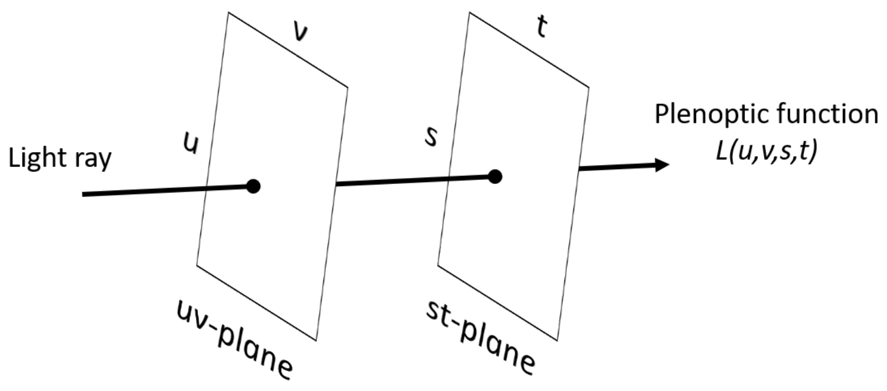
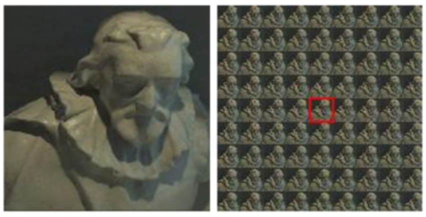


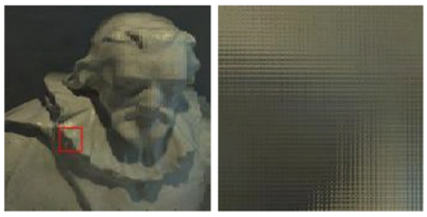
FIGURE 2.1: Schematic representation of the plenoptic function [1].

function. However, the 5D function still contains some redundancy and can be further reduced to a 4D function by considering that the radiance for a light ray travelling in free space remains constant along a straight line [36]. As shown in Fig. 2.1, the most common way to represent the 4D plenoptic function is to parameterize the light rays as an intersection of the light ray on two planes placed at arbitrary positions [16]. The plenoptic function can thus be defined as $L\ (u,\ v,\ s,\ t)$; where $(u,\ v)$ and $(s,\ t)$ denote the position of intersection of the light ray on the two planes, respectively. The function $L(u,\ v,\ s,\ t)$ is a two-plane model of light field in which the st plane can be considered as a set of cameras that have the focal plane on the uv plane [16]. The two-plane model can be interpreted in three ways as shown in Fig. 2.2.

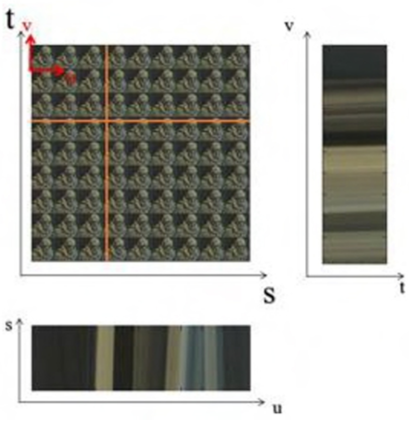
Firstly, consider that each camera captures the light ray that leaves the uv plane and arrives on the st plane, rendering an array of images that may be considered as being captured from different viewpoints. These images in the array are known as sub-aperture images or pinhole views [16] as shown in Fig. 2.2(a). Secondly, consider that a certain point on the uv plane represents the light rays inbound on all the points on the st plane as shown in Fig. 2.2(b). The s and t dimensions are known as the angular resolution, as they depend on the number of viewpoints, while the u and v dimensions are known as the spatial resolution as they depend on the camera resolution [16]. Finally, a light field image can be visualized by representing the image as an Epipolar Plane Image (EPI). The EPI can be obtained by fixing one coordinate in both spatial and angular domains,



(A)



(B)

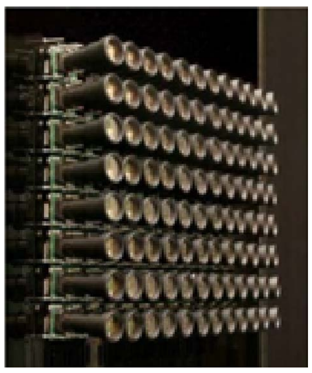


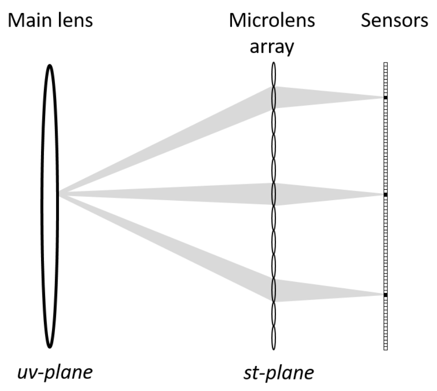
(C)

FIGURE 2.2: The three representations of the two-plane light field. (a) The sub-aperture images or the pinhole view. (b) The sub-view for the light field. (c) Epipolar-plane image obtained by fixing both spatial and angular co-ordinates [1].

while plotting pixel intensities as the other coordinates of the spatial and angular domain are varied [7]. As shown in Fig. 2.2(c), the EPI at the bottom can be represented as *E v*,t* (u, s)*, obtained by fixing the v and t coordinates, while the right EPI can be obtained by fixing u and s and represented as *E u*,s*(v, t)*. (The star symbol in the EPI signifies the fixed coordinates.) *E v,t (u, s)* refers to the EPI obtained by fixing the v and t coordinates. In this case, the EPI is represented by the intensity variation along the *(u, s)* coordinates. It means that for a fixed v and t, the EPI *E v*,t* (u, s)* shows how the intensity changes as you move along the line defined by u and s in the light field image.

Conversely, *E u,s (v, t)* describes the EPI obtained by fixing u and s coordinates. Here, the EPI *E u*,s* (v, t)* represents the intensity variations along the *(v, t)* coordinates. This means that for fixed u and s, the EPI *E u*,s* (v, t)* displays how the intensity changes when moving along the line defined by v and t in the light field image.


(A)



(B)

Figure 2.3: Video camera array to capture light field images [2] and diagrammatic representation of a plenoptic camera.

Essentially, these descriptions illustrate how EPIs capture intensity variations along specific lines or paths within the light field image. One is obtained by fixing v and t coordinates and observing changes along u and s, while the other is obtained by fixing u and s coordinates and observing changes along v and t.

2.2.1 Sub-aperture image and light field refocusing concept

The concept of sub-aperture images involves fixing the coordinates *(u, v)* while considering all variations in *(s, t)* [4]. In a light field image, the central sub-aperture picture represents the fully focused image. In Fig. 2.1, there is a conceptual model demonstrating the process of refocusing the image at a virtual focal plane *(s', t')*. To achieve this refocusing to a different focal plane *(s', t')*, the shifted versions of sub-aperture images are combined or added together.

Similarly, the creation of a focal stack involves using a filter known as the shift-sum filter. This filter works as a depth-selective filter, mimicking a planar focus. It functions by shifting slices of the light field image *(u, v)* to a common depth and then summing these slices together to generate a 2D image.

2.2.2 Light field capture

Capturing a light field can be a challenging task as a light field image captures the pixel intensity and the direction of the incident light. Two techniques that can be used to capture a light field image are shown in Fig. 2.3. An intuitive way to capture a light field is to either build a camera array with multiple cameras to capture the image from different viewpoints or by moving a camera through the scene to capture the different viewpoints. In [2], a camera array consisting of 100 video cameras was built to capture the light field image as shown in Fig. 2.3(A), whereas in [1], a video camera mounted on a computer-controlled gantry was used. However, these techniques are expensive in terms of the hardware and time taken to capture the light field images [18]. Another approach to capture a light field with a single shot places a microlens array in front of the conventional 2D camera sensor, as demonstrated in [4], [37]. This is known as a plenoptic camera. The data recorded by a plenoptic camera can be characterized using a two-plane model inside the camera given by L, where *L(u,v,s,t)* denotes the light traveling along the ray that intersects the main lens at *(u, v)* and the microlens plane at *(s, t)* as shown in Fig. 2.3(B) [4]. However, there is a trade-off between the angular and spatial resolution with the hand-held plenoptic camera that leads to a considerably lower spatial resolution compared to traditional 2D cameras [32]. But the plenoptic camera has two main advantages over a multi-camera array: first, a single capture can obtain the light field image in a plenoptic camera; second, due to gaps between cameras, the virtual aperture causes incomplete sampling resulting in the focused images from a multi-camera array to exhibit high aliasing in blurred regions [4]. Once the light field image is captured, raytracing can be applied to filter out the light rays that do not contribute to a particular depth. Rearranging the rays then estimates where the light rays would terminate if the camera was focused on the desired depth [4].

Chapter 3

Literature review

3.1 Depth estimation

Current key approaches to light field depth estimation techniques include stereo based matching, using epipolar images, depth from defocus and, more recently, the use of Convolutional Neural Networks (CNNs).

3.1.1 Depth estimation using stereo matching

Many of the proposed stereo matching algorithms are based on graph cuts and energy minimization techniques [38]. Different constraints are used to optimize the energy minimization, thus improving the estimated depth. In stereo-matching depth estimation algorithms, the goal is to assign a depth label to each pixel that is piece-wise smooth and consistent with observed data. The depth in an image tends to vary smoothly on the surface of objects but changes drastically at depth boundaries. The problem of labeling each pixel in an image with a depth value can be formulated in terms of energy minimization [38]. The aim is to find a label 'f', that minimizes the energy given the Eq. (3.1), where E_{smooth} is the measure that 'f' is not piece-wise smooth, while E_{data} measures the dissimilarity between 'f' and the observed data.

$$E(f) = E_{smooth}(f) + E_{data}(f) \tag{3.1}$$

E_{data} is given by the Eq. (3.2),

$$E_{data}(f) = \sum D_p(f_p) \tag{3.2}$$

where D_p measures how well the label f_p fits for pixel p with the observed data [38]. E_{smooth} for energy minimization that uses regularization-based approaches tend to make f smooth everywhere, which makes it unreliable at object boundaries. Discontinuities preserving the energy minimization function do not have the same issue. Graph-cut is a technique to construct a specialized graph for minimizing an energy function so that the minimum cut on the graph also minimizes the energy globally or locally [39]. In their work, Kolmogorov and Zabih [39] precisely characterize the class of energy functions that can be minimized via graph cuts. Kolmogorov and Zabih [40] also formulate the energy minimization of multi-camera scene reconstruction by following three specific criteria: treating the input images symmetrically, handling visibility and imposing spatial smoothness while preserving discontinuities. This means that the energy minimization equation contains constraints that deal with smoothness and the visibility terms. The energy that they minimize is given by Eq. (3.3)

$$E(f) = E_{data}(f) + E_{smoothness}(f) + E_{visibility}(f) \tag{3.3}$$

For the data term D, they compute intensity intervals for each color space (R,G,B) using four neighbors to then take the average data penalty [40]. For graph construction, they select a disparity α, then find the unique configuration within a single α-expansion move. Accordingly, if this decreases the energy, the algorithm goes there, or else it stops iterating. Their initial configuration satisfies the visibility constraint, ensuring all subsequent configurations will also satisfy this constraint.

Woodford *et al.* [41], on the other hand, combines visibility reasoning and second-order priors for the smoothness term for effective optimization. Their energy function is given by Eq. (3.4), where DP is the disparity, I_0 to I_N is the set of $N + 1$ input images, and I_0 is the reference view.

$$E(DP|I_0, ...I_N) = E_{photo}(I_1, ...I_N|DP, I_0) + E_{smooth}(DP|I_0) \tag{3.4}$$

Their E_{photo} term that is the data likelihood contains a visibility flag. When the input sample is visible in the reference view, its likelihood is calculated using a noise model based on a contaminated Gaussian model. In standard stereo, the smoothness prior is used to regularize the disparity map by putting a cost on unlikely geometry. The first-order derivative of disparity permits fronto-parallel surfaces i.e., the surfaces or regions in a scene that are parallel to the image plane, with no penalty. They don't assume that surfaces in the scene are fronto-parallel; they use the second derivative of disparity, allowing all planar surfaces without penalty.

Bleyer *et al.* [42] present a surface-based representation in which each pixel is assigned to a 3D surface. They then optimize an energy minimization term that takes into consideration the pixel appearance, global MDL constraint, smoothing, soft segmentation, surface curvature and occlusion. The energy function for Bleyer *et al.* [42] is given by Eq. (3.5).

$$E(f) = E_{data}(f) + E_{smooth}(f) + E_{seg}(f) + E_{curv}(f) + E_{mdl}(f) \tag{3.5}$$

Their data term E_{data} assigns a fixed penalty for occluded pixels while calculating the dissimilarity for all visible pixels. Their occlusion definition extends the asymmetric occlusion model used in [41, 43, 44], which employed the visibility constraint such that the lower disparity is occluded if the two pixels from one stereo-pair image project to the same pixel of the other pair. It was later shown in [45, 46] that this constraint doesn't hold for slanted surfaces. They state that if two or more pixels from the one stereo-pair image lie on the same surface, they can have the same matching points in the other pair without any of the pixels being occluded. This happens due to the difference in the surface sampling in the two images. Their smoothness term E_{smooth} encourages neighboring pixels to lie on the same 3D surface, and doesn't penalize based on the image edges or segment borders. The monocular information is then incorporated into the model through the soft segmentation term E_{seg}. An arbitrary segmentation divides the left image into disjoint regions, which is given as an input to the soft segmentation term. In the implementation of their work, they employ a commonly-used mean-shift segmentation algorithm, but any color or texture segmentation algorithm can be used. Their segmentation term creates a subsegment at each pixel by centering a square window at each pixel and assigns no costs if there is only a single surface within a subsegment. The curvature term E_{seg} regularizes the shape of B-spline surfaces by enforcing low curvature, and the prior is implemented as a simple unary term in the graph. The MDL term E_{mdl} penalizes on the occurrence of a surface to minimize the number of surfaces, using the MDL (Minimum Description Length) principle. Finally, the fusion move approach optimizes the model.

Yu *et al.* [47] develop an algorithm that encodes the 3D line constraints into a light field stereo matching, a line-assisted graph-cut algorithm that adds a line constraint term in the energy function that accounts for occlusion consistency. Their energy function is given by Eq. (3.6), where $E_{conventional}$ is the energy function shown in Eq. (3.3) consisting of E_{data}, $E_{smoothness}$ and $E_{occlusion}$, and they add the fourth term to Eq. (3.3), E_{line}, that is the line constraint term.

$$E(f) = E_{conventional}(f) + E_{line}(f) \tag{3.6}$$

Their critical inference for the E_{line} term is that the disparity labels to the two endpoints are allocated by checking occlusion consistency for all points along that line. The work by Boykov *et al.* [38] and Kolmogorov and Zabih [40] show that the energy function, $E_{conventional}$, can be minimized by consecutively solving the two-label problem. The two-label problem states that the algorithm at each iteration decides if the disparity should switch to the new disparity for each pixel. They use the same convention, as $E_{conventional}$ is a regular energy function since E_{data}, E_{smooth} and $E_{occlusion}$ are all two-variable functions. The energy function, therefore, can be minimized as a two-label problem with alpha expansion [38]. But, since E_{line} is a three-variable term that consists of the two endpoints and the intermediate point chosen to relabel, it can be viewed as a general second-order smoothness prior. Thus, alpha-expansion cannot be used directly to minimize E_{line}. Instead they use an extended QPBO (quadratic pseudo-boolean optimization) approach proposed by Rother *et al.* [48] to minimize the energy function E_{line}.

Chen *et al.* [22] trace all the rays passing through every 3D point to the camera array and construct an angular sampling image called the surface camera. The surface camera or SCam representation determines the radiance samples from which the correspondence and depth information can be estimated. The data structure collects all the light field rays associated with a 3D point correspondence and identical to the pinhole camera view at that correspondence stores the rays through that 3D point [49]. They show that the information from the SCam can be further analysed to distinguish occlusion profiles, introducing a Bilateral Consistency Metric (BCM) that estimates the probability at each pixel in the SCam to estimate the depth; the BCM is similar to the bilateral filters on color and spatial proposed in [50]. The disparity of the pixel is estimated by representing the consistency and disparity label as a C-D graph, showing that this metric reaches a local minimum at the correct depth for textured and occlusion cases. For occluded pixels with similar colors, they include local and global confidence metrics to test the consistency of depth estimation for these complex surfaces. Their work shows the unique property under the dense view assumption and the bilateral metric, which states that the consistency-depth curve has the property that always corresponds to a minimum on the curve at the ground-truth depth.

Tola *et al.* [51] propose a dense matching algorithm for ultra-high resolution image sets using the DAISY descriptors [52]. DAISY descriptors are computed by concatenating gradient histograms. They first estimate the gradients into separate orientation layers, then aggregate their magnitudes within each layer to form the orientation histograms. The descriptor can then be generated by thresholding and convolving the oriented gradients with Gaussian filters of various sizes. DAISY descriptors produce a similar invariance to the SIFT (Scale-Invariant Feature Transform) [53] or SURF(Speeded Up Robust Features) [54] histogram building. Still, it is efficiently computed for every image point in every direction, making it ideal for dense matching. Instead of using the computationally expensive energy mini- mization based on max-flow min-cut optimization, the DAISY matching score is directly used to compute the probability of the pixel having a particular depth. Then to produce a pair-wise dense point cloud, they consider the ratio of the two probability maxima along the uniformly sampled epipolar line by deciding if it is above or below the threshold. To enforce consistency and ensure that only the 3D points from the point cloud are retained, three geometric factors are used: the stereo-pair baseline, the focal length of the camera and the distance of the point form the center of the camera.

However, the stereo correspondence methods suffer from ambiguities while dealing with noisy and aliased regions and the narrow baseline for real camera LF images makes it difficult for these algorithms to solve the occlusion problem [7].

3.1.2 Depth estimation using epipolar plane images

Depth estimation using Epipolar Plane Images (EPI) is another popular approach. Each slice through the 4D light field representation can be used to analyse the EPI line structure to estimate the depth of the pixel under inspection as shown in Figs. 1.1(a) and 1.1(b). However, using basic line fitting techniques are not robust enough and the reconstruction is generally noisy [23]. Early work on EPI approaches can be found in [55]. Bolles *et al.* [55] state that if the camera is limited to a linear path and the image sequence contains a large number of closely spaced images, it is possible to transform a difficult 3D analysis into a set of straightforward 2D analyses. The 2D analysis then only involves detecting the lines in images that contain approximately homogeneous regions bounded by lines. 3D descriptors are then built by taking a dense sequence of images of a static scene. With the camera motion information, analysis of the object position and marking occlusion boundaries, and reconstruct the 3D structure with line fitting in the EPI.

Johannsen *et al.* [56] propose a method that learns the structural information from the central view using dictionary learning. This information is then used to raise the trained patches to the higher dimensional epipolar space by shifting the pixels proportional to their disparity and constructing a light field 'dictionary,' which is a group of 'atoms' with unique disparities. Every atom is a 2D 5 x 5 epipolar image patch generated from a specific center view atom using an individual disparity value. The coefficients are then analyzed to estimate the depth map. Even though their method improves in robustness and accuracy from previous work for multi-layered disparity estimation, which is not occlusion-aware, the accuracy of their method decreases at object boundaries as only disparity across the complete patch is considered [56].

Zhang *et al.* [23] use the epipolar images to build and minimize the matching cost for each pixel that accounts for the pixel intensity value, gradient pixel value and spatial consistency. The sub-angle estimation method then obtains the optimal slope of each pixel. This result is refined by classifying the pixels into two categories: reliable and unreliable, then using the information from the reliable pixels to replace the unreliable pixels while preserving the property exhibited by the real images. The proposed approach requires large numbers of densely sampled views, and the quality degrades considerably if the number of views is fewer than 20 [23].

Kim *et al.* [57] propose an algorithm by first estimating the depth of the object boundaries in the image by analysing the individual light rays instead of image patches. The interior regions that are more homogeneous are then processed using a fine-to-coarse procedure, by iteratively down-sampling the EPI. Unlike other approaches, this algorithm does not perform global optimization of any kind, hence preserving the object contours in the images. The input light field image used for the algorithm comprises 100 images captured by a Canon EOS 5D Mark II camera and a motorized linear stage; each image has a resolution of 21 MP. In contrast to the other algorithms that use light field camera images as input, they work with individual sub-aperture images with a resolution of 0.3 MP. The EPIs constructed with high-resolution images are much cleaner and denser, increasing the depth map accuracy. However, capturing high-resolution light field images can only be done using a camera grid or moving the camera.

Wanner and Goldluecke [58] obtain the local depth estimate by estimating the direction of the line in the EPI using the structure tensor for both the horizontal and vertical EPI. The local depth estimate only accounts for the local structures of the light field to estimate the depth. The depth labeling must be consistent across all cameras to meet the global visibility constraints. The constraint states that if a line is labeled to a certain depth value, it cannot be interrupted by a line with a label higher than that depth not to disobey the occlusion ordering. They set boundary conditions for the transition from one label to another to avoid occlusion errors. The local depth estimated from the horizontal and vertical EPI is then integrated using a convex optimization method to estimate a consistent depth map. Wanner and Goldluecke [59] address the problem of reflective surfaces in the image that cause irregularities in the epipolar plane. They propose the use of higher order structure tensors to overcome this problem, simultaneously computing the disparity maps in a 4D light field structure for both a planar reflective or transparent surface as well as the reflected or transmitted object in a 4D light field structure [59]. Criminisi *et al.* [60] describe an algorithm that operates in two phases, first by segmenting the EPI-volume into an EPI-tube, taking into account the pixel appearance. The EPI-tube is a portion of the EPI volume, which is the local orientation of the pixels from the central view within that volume [60]. The second phase describes these EPI-tubes using a simpler layer descriptor. Criminisi *et al.* [60] use the Canny edge operator, while Wanner and Goldluecke [58] use a structure tensor to obtain local disparity estimates by computing the weighted mean of each side of the line in the EPIs and then finding the distance between the two means [7]; however, occlusions and noise could cause the pixels on both sides of the lines to be different.

Zhang *et al.* [7] propose a spinning parallelogram operator that is used to analyse the EPI in order to estimate the depth. They first take the centre point of the parallelogram as the reference, and then different centre lines divide the parallelogram into two sections of the same size. The correct orientation for the central line, which is the correct depth, can be measured by finding the maximum distribution of pixel values on either side of the line. To avoid problem of inconsistent pixel distribution on either sides of the line, each side is considered separately. This assumption makes their depth estimation algorithm more robust to occlusions and noise compared to Criminisi et al. [60] and Wanner and Goldluecke [58]. It is claimed that their algorithm is insensitive to occlusion, noise, angular resolution, spatial aliasing and wide range of depth, but for noisy images their EPI-based technique generates outliers that severely effects the accuracy of their depth maps [14].

3.1.3 Depth estimation using defocus

Depth estimation from defocus has been studied extensively, where Schechner and Kiryati [24] compare the advantages and disadvantages of using defocus or correspondence cues. Research on depth from defocus also extends beyond using LF images, estimating the depth map using a single image [61, 62].

Nayar and Nakagawa [63] demonstrate that the defocused imaging system plays the role of a low-pass filter to estimate depth using the measure of focus between image points. With the same concept of shape from defocus described in Nayar and Nakagawa [63], the amount of blurring necessary to transform a sharp image to a blurred image that is an accurate representation of blur caused by a camera system depends on the depth of the scene and this can be used to estimate the depth image without recovering the deblurred image [64].

Lin *et al.* [65] state that the non-occluded pixels at different focal depths exhibit symmetry with the pixels in the in-focus slice. First estimated is the color symmetry of the focal stack with the reference central view pixels. For this, they integrate the radiance information from the pixels in the sub-aperture views that contribute to that pixel value in the refocused image and check for color consistency with the central view. They show that only a few rays contribute to the refocused image for occluded regions in the image. In contrast to stereo matching algorithms that use the color difference between the corresponding input images as a data consistency measure, the focal stack is synthesized with the hypothesized depth map and compared with the focal stack generated for the light field image data. They argue that the light field images captured by the light field camera have significant noise due to low exposure making the correspondence measure between sub-aperture views unreliable. Showing that this symmetry property is robust for noisy and undersampled light field images, as even though noise affects the individual pixel value in the sub-aperture views, the focal stack is generated by the integration process that averages out the noise.

On the other hand, Strecke *et al.* [5] use a partial focal stack approach to solve the occlusion problem. Previously, the approach proposed by Lin *et al.* [65] does not give desired results at occlusion boundaries as the foreground pixels smear into the background pixels while focusing on the background region. Lin *et al.* [65] solve this problem by choosing alternate costs for occluded pixels from the information gathered by the estimated occlusion map. But, since Lin *et al.* [65] use the estimated occlusion map, it is prone to error. Strecke *et al.* [5] propose using the light field data instead as it is not prone to estimation errors. So instead of creating a focal stack using all sub-aperture images, Strecke *et al.* [5] by using only the views right, left, above and below the central reference view, create four separate focal stacks. They assume that if the baseline is small enough and occlusion is present, it will occur only in a single direction of viewpoint shift.

Sahay and Rajagopalan [66] propose a method to obtain a high-resolution image and a high-resolution depth map using the defocus information from a stack of low-resolution images; though, the proposed algorithm is not designed for light field image depth estimation. Still, the shape-from-focus technique can also be extended for light field images, as light field images can generate focal stacks. This technique can also be used as the initial step to acquiring a high-resolution depth map and all-in-focus image from the low-resolution focal stack in the light field synthesis technique presented in Chapter 7.

Tao *et al.* [67] proposed to combine the defocus and correspondence cues using a MRF as the global optimization process. Their algorithm comprises three stages: first, using the EPI the defocus and correspondence cues are computed for depth estimation from the images; second, using these cues the optimal depth and confidence are calculated for each cue; lastly, using MRF as a global optimization process, the two are combined to estimate an accurate depth map. The limitation of this approach is the use of a fixed window size to compute the depth cues, which results in ambiguities in the depth measurement [67].

Building on the concept of combining defocus and correspondence cues for depth estimation [67], Wang *et al.* [6] take into account occlusion to estimate a more accurate depth map. To account for occlusion, an angular patch is generated for the reference pixel to then check for the photo-consistency of that patch. If the pixels in the angular patch are photo-consistent, this patch is considered to be non-occluded, and the mean and variance of the patches are computed to obtain the defocus and correspondence cues. For the angular patches that are inconsistent, the region is divided into two separate regions according to the pixel values and then the depth of these regions is computed separately. Assuming that for the occluded patches, the occluded region can only be divided into two separate regions. However, similar to other proposed techniques, the algorithm does not perform well in regions of low texture [6].

3.1.4 Depth estimation using convolutional neural networks

Convolutional Neural Networks (CNN) have been used extensively in image processing applications such as object recognition, human computer interaction, image segmentation and stereo depth estimation [25]. Over the past few years, researchers have used CNN for depth estimation in light field images [25, 26, 68]. The main concern in using CNN for light field images is that the existing light field datasets are insufficient in size to train the network, and datasets do not have accurate ground truth depth included [68].

To address this problem, Heber and Pock [68] generated a synthetic light field dataset using the raytracing software POV-Ray, Feng *et al.* [25] use the Lytro camera to capture the light field images and then use a 3dMD scanner [69] to capture the ground truth information. Shin *et al.* [8] on the other hand augment the data through scaling, center view change, rotation, transpose, and color that are suitable for light field images.

In their approach, Heber and Pock [68] extract the information from the vertical and horizontal EPIs to input into the CNN. The resultant depth map is optimised by solving a convex energy minimisation problem. Unlike Heber and Pock [68] that use only one directional EPI, Shin *et al.* [8] construct multi-stream networks for four viewpoints, horizontal, vertical, left and right diagonal directions. They show that their multi-stream network reduces the reconstruction error over a single-stream network by 10%, claiming that the use of a 2×2 kernel in the algorithm reduces the effect of noise.

In [25], the network consists of two parts: the encoder and the decoder. The encoder extracts a set of high-level feature maps for the light field inputs, and then these sets of feature maps are decoded to estimate the depth map. Then, a two stream CNN network that uses the correlations across multiple neighbourhood pixels learns to estimate the depth from EPIs. The variational model is then refined using the prior image from the central view.

Han *et al.* [70] devised the ESTNet, a convolutional neural network aimed at accurately estimating depth maps. The architecture of ESTNet is distinguished by three input streams, featuring an encoding-decoding structure and skip connections. These input streams correspond to the horizontal EPI synthetic image (EPIh), vertical EPI synthetic image

(EPIv), and central view image (CV), respectively. In the network's forward propagation, skip connections are incorporated between the convolution module and the corresponding transposed convolution module, facilitating the fusion of shallow local and deep semantic features. However, a limitation of their algorithm lies in its design for nine views in either the horizontal or vertical direction. Consequently, this method lacks adaptability to varying numbers of views. While our existing light field synthesis algorithm currently generates a 9 x 9 light field image, it is important to note that it does not rely on a CNN network. Consequently, the code can be easily modified to produce a greater number of views by making adjustments to a few lines of code.

In their research, Heber *et al.* [71] investigated a convolutional neural network for predicting the orientation of the 2D hyperplane in the light-field domain, representing the depth of the 3D scene point. Additionally, Heber *et al.* [71] formulated a convex optimization problem incorporating high-order regularization. It is worth noting that Heber's CNN does not operate as an end-to-end network for depth estimation from this perspective.

Guo *et al.* [72] further decomposed a complex task into multiple straightforward sub-tasks, each addressed by a dedicated subnetwork. They specifically divided the depth estimation into non-occlusion and occlusion regions, recognizing the distinct properties of these regions in relation to the light field structure—complying with and violating the angular photo consistency constraint, respectively. Their network incorporates three modules: the Occlusion Region Detection Network (ORDNet), the Coarse Depth Estimation Network (CDENet), and the Refined Depth Estimation Network (RDENet). In detail, ORDNet predicts the occlusion map as a mask. Guided by this occlusion map, CDENet focuses on depth estimation in non-occlusion areas, while RDENet handles depth estimation in occlusion areas. While the outcomes showcased in the paper appear encouraging, it's crucial to highlight that the network is exclusively trained on synthetic images. The training dataset comprises 38 images sourced from the Dense Light Field Dataset (DLFD) [73] and an additional 16 images obtained from the 4D light field dataset [3]. It is notable that the network is neither trained nor its results demonstrated on real light field images.

Leistner *et al.* [74] addressed the challenge of light field depth estimation in high-resolution and wide-baseline light fields using neural networks. Rather than expanding the receptive field of the network, which might lead to poorer generalization, they suggested a different approach: transforming the input images through a shear transformation named EPI-Shift. This method involves multiple forward passes with shifted input EPIs, and the results are then combined into a single prediction. Regrettably, the U-Net model appears to be susceptible to noise, particularly at object boundaries that do not align with the disparity label boundaries. In contrast, our algorithm exhibits robustness to noise when it comes to depth estimation.

Chen *et al.* [75] strive for enhanced utilization of light field data through the analysis of horizontal, vertical, and diagonal EPIs. However, directly fusing all features from these EPIs can lead to ambiguous cost volumes in occluded regions, making it challenging to determine correct disparities. To address this issue, they introduce the Intra-Branch Fusion module, where features are fused within each branch. This fusion helps prioritize views on one side that are less likely to contain occluded regions. The information obtained from the four branches is judiciously merged, with branches possessing clearer matching information contributing more to the cost volume. Instead of a simple concatenation operation, they devise the Inter-Branch Fusion module to blend features from different branches. This involves assigning larger weights to branches with fewer occlusions and clearer correspondences. A limitation of this approach lies in the manual removal of reflection, refraction, and texture-less areas during training to maintain the consistency of matching. While this is feasible for smaller training datasets, such as the 16 images utilized in this study, it could become impractical when training the network on a significantly larger dataset.

Similar to the approach by Chen *et al.* [75], in Tsai *et al.* [76], the goal is to maximize the utilization of light field data by using all 81 sub-aperture views as input and generating a depth map for the centre view.Each sub-aperture view of the input light field image undergoes four basic residual blocks. In a residual block, the neural network concentrates on learning the residual (difference) between the input and the output, instead of directly mapping the input to the output. The output of the residual block is derived by summing the input and the learned residual [77]. The feature map obtained for each view is then inputted into a Spatial Pyramid Pooling (SPP) module, which extracts contextual information from the scene. The SPP module divides the image into partitions from finer to coarser levels, aggregating local features in each division [78]. To discount redundant information from certain views, an attention-based sub-aperture view selection module is employed to learn the importance of each view. The attention map is added to the estimation module, allowing a focus on more crucial views and enhancing overall accuracy. To prevent erroneous correspondences during training on the 4D Light Field Dataset, we eliminate patches from regions containing objects with non-diffuse reflection and refraction, such as glass, metal, and textureless areas. However, this exclusion process might pose practical challenges when scaling up the training of the network to a substantially larger dataset.

Peng *et al.* [79] present a zero-shot learning-based framework for light field depth estimation. This framework learns an end-to-end mapping exclusively from input light fields to corresponding disparity maps without the need for additional training data or supervised ground truth depth. Their CNN comprises four stages. In the first stage, Feature CNN, two features are extracted across shifted sub-aperture images at various disparity levels, representing mean and standard deviation features. The second stage, Disparity CNN, computes the disparity map based on the extracted depth feature. To train the network without ground truth depth labels, a combined loss inspired by the photo-consistency constraint in [6] is designed. This new loss function, distinct from the one used in the earlier unsupervised network [80], comprises two terms: compliance and divergence, leveraging spatial and angular correlations in the 4D light field to constrain the network. The third stage, Warp Layer, warps sub-aperture images to the central view using the generated disparity, and the Combined Loss is employed to update parameters in both Feature CNN and Disparity CNN. This iterative process continues until the network converges. A drawback of this algorithm is its setting of the number of disparity levels (N) to 100, striking a balance between accuracy and speed. Consequently, the precision of the depth map becomes dependent on the depth levels of the light field image. Light field images with less depth range yield depth maps with greater precision than those with a larger depth range, leading to inconsistent results.

Alperovich *et al.* [81] introduce a fully convolutional autoencoder designed for light field images, capable of extracting disparity, diffuse, and specular intrinsics. While learning-based approaches show promise in depth estimation from light field data, they typically rely on abundant training data with supervised ground truth depth. Obtaining such data in real-world scenarios poses challenges. Furthermore, learning-based methods often encounter domain shift issues when generalizing to significantly different inputs. A supervised model well-trained on a synthetic dataset with ground truth depth may prove fragile when applied to real-world scenes. Unsupervised networks offer an advantage by training without the need for ground truth depth labels, making them more practical for real-world scenes. However, they still require an additional training dataset, exposing them to potential domain shift effects in unsupervised learning.

3.2 Light field image synthesis

View interpolation is the process of estimating intermediate views of the scene given a set of images of the scene from different viewpoints. This is a well-known method for image synthesis using a set of reference views. In one of the earlier works on image synthesis [82], pixel correspondences are established using the range data and the camera transformation parameters between a pair of two consecutive images and a pre-computed morph map is used to interpolate intermediate views. In [63], it is noted that the holes that are generated in the estimated intermediate views as the foreground regions in the estimated views move more than the background regions. These holes are filled by interpolating the adjacent pixels near the holes, which causes the filled regions to be blurry.

Light field synthesis approaches either use a sparse set of perspective views to synthesise the view inside of the image baseline or use a focal stack or the central view to extrapolate the perspective views. Based on the current approaches, light field synthesis can be classified into two main categories, non-learning based and learning based approaches. In non-learning based approaches, Mousnier et al. [17] use gradient detection and graph-cut to determine the focus map, then using the camera metadata convert the focus map to a depth map. The tomographic reconstruction of the epipolar images by back-projecting the focused regions thus synthesise the light field. In contrast, Lui et al. [83] reconstructed the light field via filtered-back-projection and the Simultaneous-Algebraic-Reconstruction-Technique (SART) algorithm.

Many learning based techniques also use focal stacks for LF synthesis [80, 83–85]. Srinivasan [84] propose a light field synthesis algorithm using only the central all-in-focus image. The CNN- based algorithms use the light field images to generate the focal stack and then use the focal stack as inputs and train the network using the full light field image. Two drawbacks of these algorithms are: The dataset used does not have a ground-truth depth map, and a 2D camera does not capture the focal stack for the light field synthesis.

3.2.1 Non-learning-based light field synthesis

Kubota *et al.* [31] use a focal stack captured from multiple viewpoints to synthesise intermediate views. Assuming that the scene has only two focal regions, a background and a foreground, the inputs for their approach are four images: two images captured by each camera for the background and foreground regions. The drawback of the approach is that it requires images to be captured from two viewpoints, which is a complex set-up and the resultant synthesised image only has two focus planes. In their work, Mousnier *et al.* [32] propose partial light field reconstruction from a focal stack. They use the focal stack images captured by a Nikon camera to estimate the disparity map and an all-in-focus image, then use the camera parameters to estimate the depth map. They use the depth map and all-in-focus image to synthesise only one set of nine horizontal and nine vertical perspective views, but since the algorithm requires data from the camera parameters it is difficult to run the code on a light field image to check the accuracy against light field sub-aperture images.

Levin *et al.* [28] also use focal stack images but instead of using depth estimation for the synthesis, show that using a focal stack the 4D light field can be rendered in a linear and depth invariant manner. They argue that a focal stack is a slice of the 4D light field spectrum and thus the focal stack directly provides the set of slices that comprise the 3D focal manifold that can be used to construct the 4D light field spectrum. But their dimensionality gap model is unreliable at depth boundaries resulting in the background pixels linking into the foreground pixels.

Perez *et al.* [86] propose a light field recovery algorithm from its focal stack that is based on the inversion of the Discrete Focal Stack transform. They show that the inversion using the Discrete Focal Stack transform requires many images in the focal stack. For a standard plenoptic camera featuring 255×255 microlenses and 9×9 pixels behind each microlens, the utilization of the inversion formula would necessitate capturing a focal stack comprising 2033 images. They then show practical inversion procedures for general light fields with various types of regularizers, such as L2 regularization of 0th order and 1st order, and L1 isotropic TV regularization. By incorporating L2 regularization of 0th and 1st order, along with L1 isotropic Total Variation (TV) regularization, penalties are imposed based on the weight's magnitudes, their first-order derivatives, and the L1 norm of the image gradient. This combined regularization strategy aids in mitigating overfitting, fostering sparsity, and facilitating smoother solutions within optimization problems. The two main drawbacks of the algorithm proposed by Perez *et al.* [86] are that inversion using the Discrete Focal Stack transform requires a large number of images in the focal stack, and they need to use regularization approaches to stabilize the transform.

Zhang *et al.* [9] in their work use one micro-baseline image pair to synthesise the 4D light field image. It is proposed that the small-baseline image pair can be captured using vibration in a static camera or by slight movement of a hand-held camera. There are two limitations of the approach: first, that the depth estimation algorithm used reduces in accuracy as the distance between the input views is increased; second, that since no information is available to fill in the gaps generated by the difference in movement of the background and foreground regions in the sub-aperture images, the accuracy of the edge sub-aperture images is reduced considerably compared to the sub-aperture images closer to the central view.

Shi *et al.* [27] in their approach use a sparse set of light field views to predict the views inside the boundary light field images used, but since the approach requires a specific set of sub-aperture views as input data from the light field images, applying the algorithm to different types of data is non-trivial. The approach can be used for applications like light field compression, but not for light field synthesis as it requires a set of sub-aperture views as input data.

3.2.2 Learning-based light field synthesis

Kalantari *et al.* [10] propose a two network learning based light field synthesis approach that uses a sparse set of four corner sub-aperture images to estimate the depth map with the first network, then the second network estimates the missing RGB sub-aperture images. Gul *et al.* [87] propose a three-stage learning-based light field synthesis approach that also uses a sparse set of four corner sub-aperture images. The first stage is the stereo feature extraction network, the second stage is a depth estimation network, and the third stage uses the depth map to warp the input corner view to have them registered with the target view to be synthesized. One drawback of the proposed algorithm is that capturing the four corner sub-aperture images is not straightforward, and would either require moving the camera manually or a special apparatus with multiple cameras. However, the approach can be used for light field compression as the approach uses corner sub-aperture views as input data.

Srinivasan *et al.* [33] propose a CNN that estimates the geometry of the scene for a single image and renders the Lambertian light field using that geometry. The second CNN stage then predicts the occluded rays and non-Lambertian effects. The network is trained on a dataset containing 3300 scenes of flowers and plants captured by a plenoptic camera. But since the algorithm predicts the 4D light field image using a single image, filling the regions in the sub-aperture image at large discontinuities will only be an approximation as that information isn't available from a single image. They extend their network by training it on 4281 light fields of various types of toys including cars, figurines, stuffed animals, and puzzles, but their results show that the images are perceptually not quite as impressive as the images synthesised for the flower dataset.

Wang *et al.* [88] propose a two-stage position-guiding network that uses the left-right stereo pair to synthesize the novel view. They first estimate the depth map for the middle/central view and then check the consistency for the synthesized left and right view. The second CNN is the view rectifying network. They train their network on the Flyingthings3D dataset [89] that contains 22,390 pairs of left-right views and their disparity maps for training and 4370 pairs for testing. The main limitation of the approach is that, since their research focuses on dense view synthesis for light field display, they only generate the central horizontal views and not the entire light field image.

Wu *et al.* [12] present a “blur restoration-deblur” framework for light field reconstruction using EPIs. They first extract the low-frequency components of the light field in the spatial dimensions using a blur kernel on each EPI slice. They then implement a CNN to restore the angular details of the EPI, and they use a non-blind “deblur” operation on the blurred EPI to recover the high spatial frequencies. In their work, they also show the effectiveness of their approach on challenging cases like microscope light field datasets [12]. The main drawback of their approach is that they need at least three views for both angular dimensions for the initial interpolation, and their framework cannot handle extrapolation.

Yeung *et al.* [13] propose a learning-based algorithm to reconstruct a densely-sampled light field in one forward pass from a sparse set of sub-aperture views. Their approach first synthesises intermediate sub-aperture images with spatial-angular alternating convolutions using the characteristics of the sparse set of input views, and they then use guided residual learning and stride two 4D convolutions to refine the intermediate sub-aperture views. They suggest that the proposed algorithm can be used for light field compression and also applications such as spatial and angular super-resolution and depth estimation. In their 2 x 2 - 8 x 8 set-up where they use four sub-aperture views as input, Yeung *et al.* [13] extrapolate only two views in both directions, with the remaining views being interpolated within the baseline of the input images. In contrast, our approach involves using only the central image as input and extrapolating three images in both directions to synthesize a 7 x 7 light field image.

Zhou *et al.* [90] in their work train a deep network that predicts the multi-plane image using an input stereo image pair. A Multi-Plane Image (MPI) is a set of images where each plane encodes the RGB image and an alpha/transparency map at each depth estimated by the stereo image pair. The MPIs can be considered as a focal stack representation of the scene, predicted using only the stereo image as input. If the stereo base-line is large enough, the parts of the image that are visible due to the lateral shift can give information that can be used to fill in the gaps generated by the difference in region depths in the perspective views in the horizontal direction.

Chao *et al.* [11] propose a lightweight CNN that uses a single stereo image pair that enforces the left-right consistency constraint on the light fields synthesized from left and right stereo views, and then merges the light field synthesized by right and left stereo views using a distance-weighted alpha blending operation. But since the input stereo pair used is only in the horizontal direction, gaps in the vertical perspective views can only be filled by using a prediction model as no information is available in the vertical direction.

Traditional methods for rendering 3D scenes typically involve the creation of explicit geometric models representing objects and their surfaces. These models can become intricate and computationally demanding, especially when dealing with scenes featuring detailed elements or dynamic components. In contrast, Mildenhall *et al.* [91] introduce NeRF which offers an alternative approach by directly modelling the volumetric scene function using neural networks. NeRF represents a 3D scene as a continuous volumetric function that describes the radiance (color and light intensity) at any given point in space. Training NeRF involves using a set of images captured from various viewpoints. Despite its impressive results in generating realistic 3D scenes, NeRF can be computationally intensive, particularly during the training phase.

In their study, Zhang *et al.* [92] pinpointed a fundamental ambiguity in the shape radiance of NeRF models, which arises due to training exclusively with a photometric loss instead of incorporating supervised learning with ground truth depth. In response to this observation, they propose NeRF++ as an extension of the original Neural Radiance Fields (NeRF) approach, with the goal of overcoming limitations and improving the practicality and efficiency of the method. This discrepancy allows a family of radiance fields to accurately account for all training images, even when the underlying shape is incorrect. NeRF++ tackles the computational complexities associated with the original NeRF by introducing a sparse neural representation that selectively employs a subset of 3D points for both training and inference. This modification leads to a more robust and scalable framework for acquiring detailed 3D scene representations from images. However, it is crucial to note that the NeRF methods are tailored for 3D reconstruction based on a collection of images captured in 3D space, not explicitly for light field images. While there is potential for extending these techniques to light field reconstruction, such an extension falls outside the scope of our current work.

3.3 Summary

The approaches reviewed in this chapter estimate the depth for light field images using stereo correspondence, EPIs, depth-from-defocus, and CNNs. The stereo-based algorithms suffer from ambiguities while dealing with noisy images, and the narrow baseline for real camera LF images makes it difficult for these algorithms to solve the occlusion problem [7]. The EPI-based algorithms claim to be insensitive to occlusion, noise, angular resolution, spatial aliasing and wide range of depth. However, for noisy images, the EPI-based techniques generate outliers that severely affect the accuracy of their depth maps [14]. The algorithms that use depth from defocus cause ambiguities in the depth map near large depth discontinuities. For the CNN-based algorithms, they generally suffer in accurately evaluating the algorithm accuracy due to insufficient size to train data, satisfying all imaging conditions and exclusion of ground truth depth maps [68].

The light field synthesis algorithms reviewed in this chapter are divided based on learning and non-learning-based algorithms. Algorithms that use stereo-image pairs as input for synthesis produce inaccuracies at depth discontinuities as no information is available to fill the gaps generated in the vertical views.

Our research focuses on depth estimation algorithms for light field images using focal stack images to improve the accuracy around depth discontinuities and noise resilience. To solve the inverse problem of light field image synthesis using the focal stack and depth map, improving the filling of occluded regions in the synthesized views.

Chapter 4

Light field toolkit for generating focal stack images

4.1 Introduction

In this chapter, we address the challenge that existing light field datasets being used for light field synthesis do not contain focal stack images. The toolkit presented in this chapter extends on the work of Honauer *et al.* [3], and allows researchers to create datasets containing light field images, corresponding ground-truth depth map, disparity map, and focal stack captured by a 2D camera model. This enables the evaluation of algorithms that use depth estimation as an intermediate step for light field synthesis, to test the accuracy of both the depth estimation and light field synthesis. In the rest of the thesis, this toolkit is used to analysis and validate our proposed approaches for depth estimation and light field synthesis presented in Chapters 5, 6, and 7.

4.2 Existing light field datasets

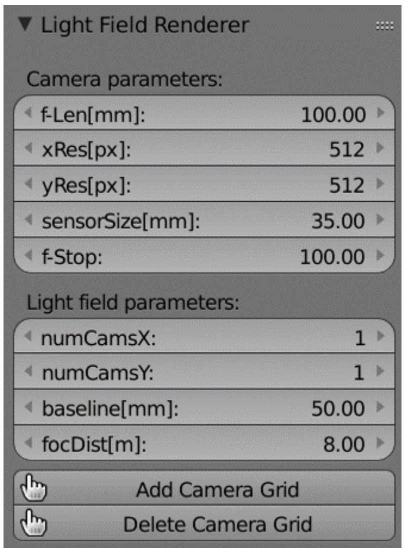
Light field datasets can be divided into synthetic and real datasets. Synthetic datasets are created using 3D modelling and rendering software such as Blender [93]. Real-world light field images are captured by either using a plenoptic camera, a camera array, or gantry. A few existing light field datasets are: the new Light Field Image Dataset [94], the 3D High-Resolution Disney Dataset [57], the Stanford Light Field Archive [2], and the Synthetic Light Field Archive [95]. However, these datasets do not contain ground-truth depth maps, making it difficult to evaluate algorithm accuracy. On the other hand, HCI Light Field Benchmark [96] and 4D Light Field Dataset [3] contain the light field images and the ground-truth depth map. However, none of these datasets contain the focal stack of the same scene as the light field images. The lack of focal stack images alongside the corresponding light field images of a scene poses challenges for light field image synthesis algorithms. Without these focal stack images, it becomes challenging to compare the algorithm-generated results with the original light field images. This lack of comparative data hampers the evaluation of algorithm accuracy and performance.

4.3 Prior Work

The 4D Light Field Dataset [3] created a light field camera model in Blender [93] that can render light field images with variable camera parameters. Specifically, they created a light field camera simulation model using Blender. In this chapter, we extend on their work and create a model to enable researchers to create focal stack images for the same light field scene.

The model in [3] lets the user create a light field camera with different camera parameters. The camera parameters include the focal length, image resolution, sensor size, and f-stop number. These values are updated for all the cameras in the light field model. Users can also vary the light field parameters, such as the number of cameras, the distance between consecutive cameras, and the focus distance of the cameras; Fig. 4.1 shows the light field model graphical user interface with default values. The model also lets the user visualize the image space in terms of a frustum, making it convenient for setting up the objects in the scene.

The light field image can be rendered once the camera parameters and the scene are set. The rendering generates an 8-bit sub-aperture image of the user-defined resolution for each



Light Field Renderer

Camera parameters:

f-Len[mm]: 100.00

xRes[px]: 512

yRes[px]: 512

sensorSize[mm]: 35.00

f-Stop: 100.00

Light field parameters:

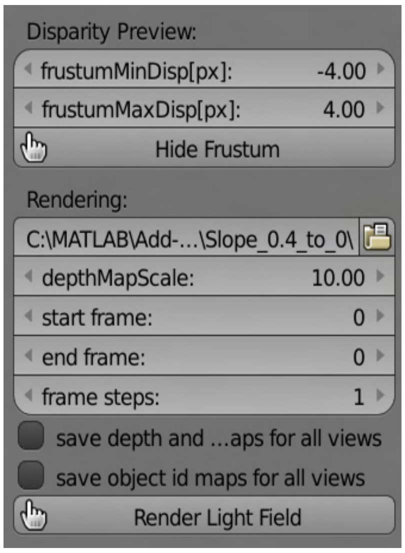
numCamsX: 1

numCamsY: 1

baseline[mm]: 50.00

focDist[m]: 8.00

(A)



Disparity Preview:

frustumMinDisp[px]: -4.00

frustumMaxDisp[px]: 4.00

Rendering:

C:\MATLAB\Add-...\Slope_0.4_to_0\

depthMapScale: 10.00

start frame: 0

end frame: 0

frame steps: 1

save depth and ...aps for all views

save object id maps for all views

(B)

FIGURE 4.1: Light field parameters with default values as seen in the Blender interface for the light field camera model of Honauer et al. [3]

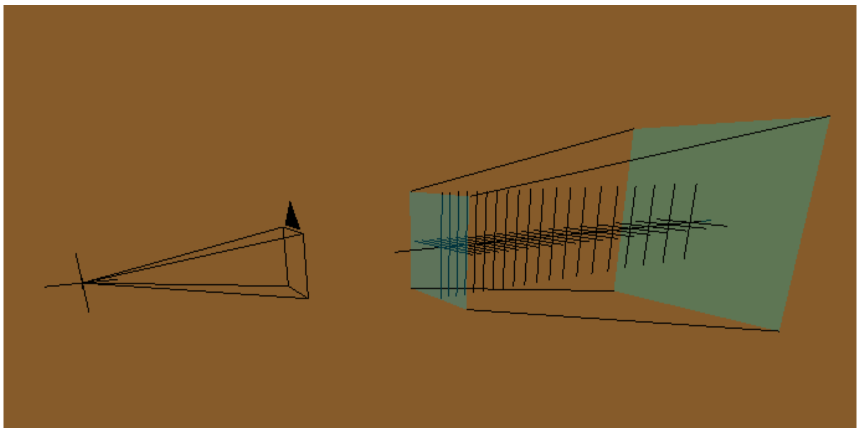


FIGURE 4.2: Sample focus points, as seen in the Blender interface for our proposed toolkit.

camera in the model. By default, the model generates an 8-bit light field of size (9 × 9 × 512 × 512 × 3). Then, the light field render generates 16-bit ground truth disparity and depth maps in two resolutions (512×512 px and 5120×5120 px) for the central view, but can also create disparity and depth maps for all sub-aperture views. The model also generates a 'parameter.cfg' file that lists all the camera parameters used for the light field image.

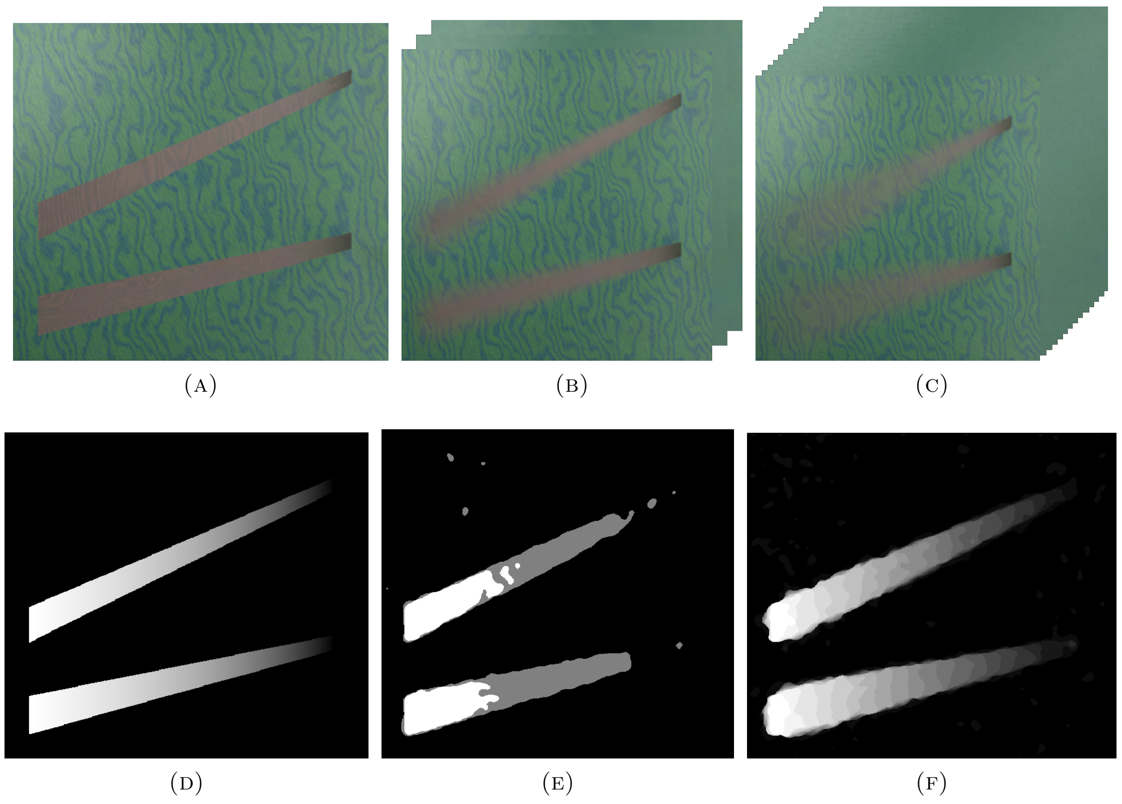
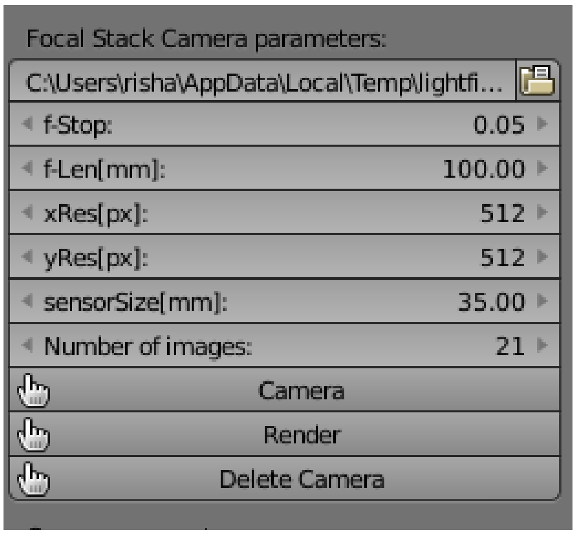


Figure 4.3: (a) The all-in-focus or central image . (b) Focal stack with 3 images. (c) Focal stack with 21 images. (d) Ground-truth depth map. (e) Depth map using 3 focal stack images. (f) Depth map using 21 focal stack images.

4.4 Our Work

We extend the work of Honauer et al. [3] by creating a camera model to generate a focal stack and an all-in-focus image for the same scene as the light field image. Our focal stack rendering framework is publicly available on GitHub [1] as the Focal Stack Toolkit. The model lets the user create a focal stack with different camera parameters that include the focal length, image resolution, sensor size, and f-stop number. Fig. 4.4 shows the graphical user interface developed in this chapter for the focal stack camera with default values. By default, the model generates an 8-bit focal stack image with size (512 × 512 × 3), but the image resolution can be increased according to the user preference. We also create a similar parameter configuration file as in [3]: our 'parameter_fs.cfg' file lists all the camera parameters and the focus distance of the focal stack images.

[1]https://github.com/rishabhsharma27/Light-field-and-focal-stack-toolkit



Focal Stack Camera parameters:

C:\Users\risha\AppData\Local\Temp\lightfi...

f-Stop: 0.05

f-Len[mm]: 100.00

xRes[px]: 512

yRes[px]: 512

sensorSize[mm]: 35.00

Number of images: 21

FIGURE 4.4: Focal stack camera parameters with default values, as seen in the Blender interface for our proposed toolkit.

In addition to the camera parameters, the user has an option to enter the number of images they need in the focal stack. The number of images required for the focal stack determines the focus point for the camera for each image in the focal stack. We use the disparity values from the light field camera parameters and the number of images to create focus points at equal intervals. The disparity values differ from the depth values used to select the focal point. As you move away from the camera, the distance between the focus point for consecutive images in the focal stack increases. The focus points for the focal stack with 21 images is shown by the cross-hair in Fig. 4.2. The depth value in metres for each disparity value is calculated using the sensor size, focal length, distance between the consecutive light field cameras, and image resolution. The cross hairs in Fig. 4.2 are placed at a constant disparity difference of 0.4 with minimum and maximum disparity of -4 and +4, respectively. Figure 4.3 (a) and (d) show the central light field view or the all-in-focus image of the focal stack and the ground-truth depth map. Figure 4.3 (b) and (c) show an example of focal stack with three and twenty-one images respectively and the corresponding estimated disparity maps are shown in Figure 4.3 (e) and (f). Figure 4.5 (a) and (b) show the focus point for the corresponding focal stack with three and twenty-one images.

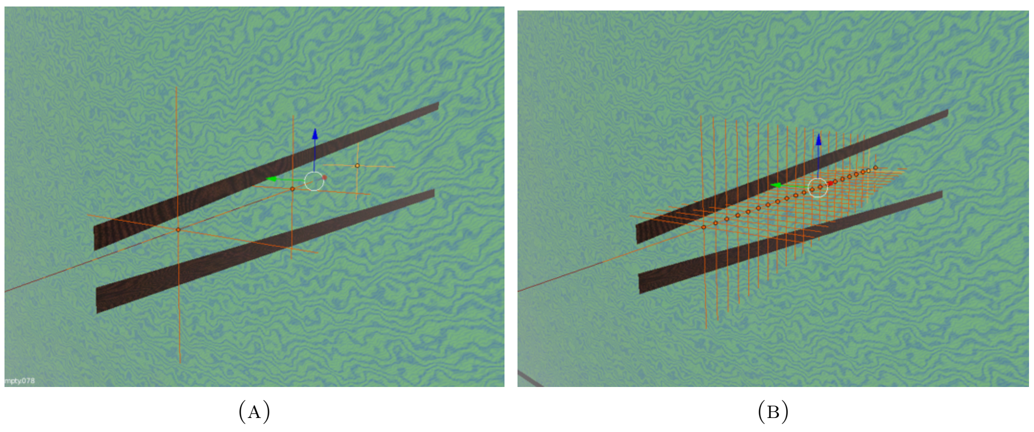


FIGURE 4.5: (a) Focal point for focal stack with 3 images. (b) Focal point for focal stack with 21 images.

4.5 Discussion and Conclusion

This chapter presented a toolkit using the freely available and open-source Blender software to generate a dataset containing the light field image, ground-truth depth, disparity map, and focal stack captured by a 2D camera model. Extending an existing published light field camera toolkit, our proposed toolkit allows researchers to generate a focal stack for the same light field image to allow light field synthesis algorithms to evaluate performance accuracy.

In this thesis, we were able to analyse the focal stack generated using our toolkit, and the focal stack generated using the Matlab toolkit [97] to understand and address the errors to improve the precision of our depth map as explained in Chapter 5 Section 5.2.1. We were also able to understand how the focal stacks could be used to fill the gap generated in the synthesised views at depth discontinuities presented in Chapter 7 Section 7.2.2. In future, we believe that our proposed toolkit can also facilitate improvements in CNN and deep learning approaches that use focal stack images for light field synthesis.

Chapter 5

Depth Estimation for Light field images

5.1 Introduction

This chapter focuses on algorithms for the estimation of depth maps for light field images using focal stack images. Existing depth estimation algorithms for light field images often fail to accurately estimate the depth around depth discontinuities due to occlusions and noisy images. To address this challenge, in this chapter we propose a depth map estimation approach that is novel in two main ways: 1) using frequency domain representations of image patches instead of the RGB patches to increase the noise resilience; and, 2) focal stack generation technique to ensure sub-pixel accuracy for the focal stack, and to increase the accuracy around occlusions and the depth precision of our depth maps.

5.2 Methodology

The depth estimation approach presented in this chapter exploits the characteristic of light field images having multiple focal planes that are captured in a single image. The flow of the algorithm is represented in Fig. 5.1, where the proposed approach is divided into

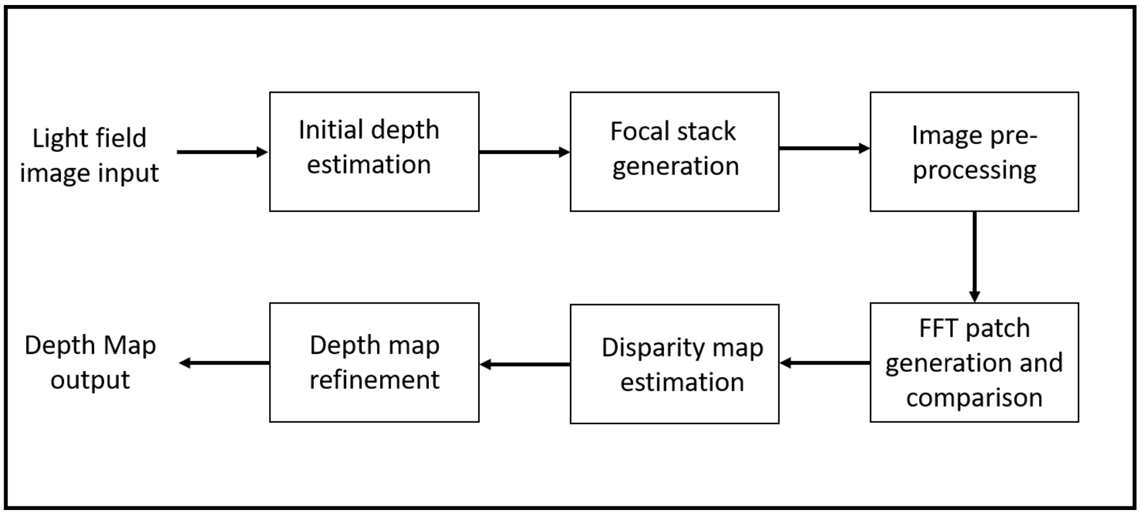


FIGURE 5.1: Flow of the proposed algorithm.

four main sections: initial depth estimation, focal stack generation, patch generation and comparison, and depth refinement. Unlike the depth from defocus algorithms described in Chapter 3, our approach uses frequency patch analysis which makes the algorithm more resilient to noise.

5.2.1 Initial depth estimation

The initial depth estimation algorithm is similar to the depth estimation algorithm used for the proposed techniques shown in Fig. 5.1, the only difference being that the initial depth estimation algorithm only determines the maximum and minimum slope and corresponding depth values in the light field image, rather than the intermediate depth values. When considering EPIs or other depth estimation techniques from a light field, the slope values within these images or data representations can often be associated with the depth of objects or points within the scene, hence slope here means the depth for the refocus plane.

In this work, we select the depth or slope difference between two consecutive focal stack images to be 0.2 for this stage. We chose a depth difference of 0.2 because it covers the range needed for accurate estimation of peripheral depth values, all while minimizing the computational time required.

The experimental evaluation also proved that this value covers the depth range at equal depth intervals to allow an accurate estimation of periphery depth values. The initial depth estimation algorithm comprises three stages: the first stage generates the focal stack images at a depth difference of 0.2, and as this step is only to determine the maximum and minimum depth values, we reduce the number of sub-aperture views to generate the focal stack. The second stage divides the focal stack and all-in-focus image into patches of smaller patches of 4x4 pixels. The third stage compares the smaller patches with the all-in-focus patches to generate a depth map. From this depth map, we only choose the maximum and minimum depth values used as a reference to generate the focal stack in the next stage of the depth estimation algorithm. The initial depth estimation stage improves the depth estimation in two ways: firstly, it reduces the computational time as only the relevant focal stack images are generated; and , secondly, it reduces the number of redundant images to reduce the possibility of misdetections.

5.2.2 Focal stack generation and image pre-processing

A single LF image can be used to generate an all-in-focus image and also to generate the same scene at different focal lengths and with a narrow depth of field. The sub-aperture images can be obtained by holding *(u,v)* fixed and considering all *(s,t)* [4]. The central sub-aperture image is the all-in-focus image. Fig. 5.3 shows the conceptual model of a light field image when refocusing at a virtual plane *(s',t')*. Thus, to refocus the image to a different focal plane *(s',t')*, the shifted versions of the sub-aperture images are summed together. Following the same concept, the focal stack is generated using the shift-sum filter, which is a depth-selective filter that functions like a planar focus. The filter works by shifting *(u,v)* slices of the LF image to a common depth and then adding the slices together to obtain the 2D image. The value of the slope controls the amount of shift that corresponds to the image being focused on a different focal plane. Fig. 5.2 shows the flow of the shift-sum filter refocusing algorithm.

Table 5.1 shows the values by which individual sub-aperture images are shifted to refocus the images to a particular focal point using the slope value using the Matlab toolkit [97]. The negative value in Table 5.1 indicates that the sub-aperture images are shifted to the

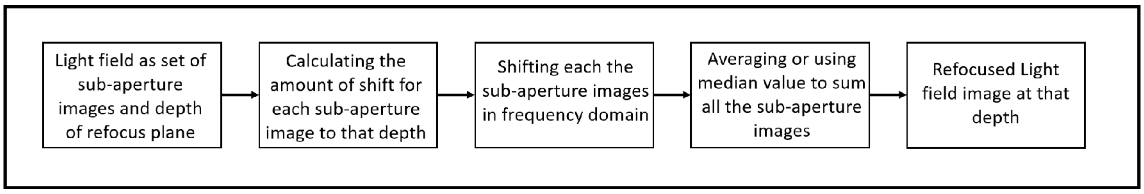

Light field as set of sub-aperture images and depth of refocus plane
Calculating the amount of shift for each sub-aperture image to that depth
Shifting each the sub-aperture images in frequency domain
Averaging or using median value to sum all the sub-aperture images
Refocused Light field image at that depth

FIGURE 5.2: Algorithm for LF image refocusing using Shift-sum filter

right and downwards, whereas the positive value indicates that the sub-aperture images are shifted to the left and upwards. In the initial implementation for light field image refocusing the u,v values were calculated using Eq. (5.1) and Eq. (5.2), where LFSize(1) and LFSize(2) are the light field image angular resolution in the horizontal and vertical direction respectively, TV slope and SU slope are the focal point at which the image needs to be refocused. The amount the sub-aperture images need to shift is a product of the slope and how far the sub-aperture images are from the central view. The 'linspace' function in Matlab$^{\text{TM}}$ is employed to evenly partition a vector along a linear range, specifically within the angular resolution of a light field image, spanning from -0.5 to 0.5; this ensures that we have an amount by which each sub-aperture view needs to be shifted in accordance with the position on the sub-aperture view in the light field image. The vector values obtained from Eq. (5.1) and Eq. (5.2) are not specific to a particular scene type but can be used with all light field images. The fact that the equations use the angular resolution to multiply with the linear range vector ensures that it can be scaled according to the angular resolution of the light field image.

As shown in Table 5.1, Eq. (5.1) and Eq. (5.2) give us the value by which each sub-aperture image needs to be shifted to allow us to focus on a particular focal point in the scene for a specific slope. While using the values obtained from Eq. (5.1) and Eq. (5.2) shown in Table 5.1 to generate the focal stack, we observed that there was an error with the estimated focal stack and that the error would increase as we focused closer or further away from the camera, i.e., near -4 and +4 slope values.

$$V_{Offset} = linspace(-0.5, 0.5, LFSize(1)) * Slope * LFSize(1) \tag{5.1}$$

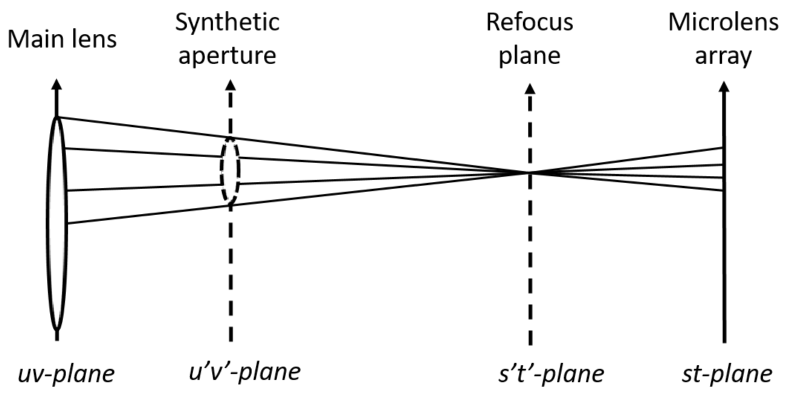


FIGURE 5.3: Conceptual model for refocusing a LF image at a different depth [4].The *(u,v)* and *(s,t)* plane are surfaces of the camera respectively and *(s',t')* is the refocus plane

TABLE 5.1: Light field refocusing shift variable from the Matlab light field toolbox.

Slope	Sub-aperture image distance from central view								
	4	3	2	1	0	1	2	3	4
	Number of pixels by which the sub-aperture images are shifted								
0	0	0	0	0	0	0	0	0	0
1	-4.5	-3.375	-2.25	-1.125	0	1.125	2.25	3.375	4.5
2	-9	-6.75	-4.5	-2.25	0	2.25	4.5	6.75	9
3	-13.5	-10.125	-6.75	-3.375	0	3.375	6.75	10.125	13.5
4	-18	-13.5	-9	-4.5	0	4.5	9	13.5	18

$$U_{Offset} = linspace(-0.5, 0.5, LFSize(2)) * Slope * LFSize(2) \tag{5.2}$$

The amount by which the sub-aperture images need to shift is a product of the slope and how far the sub-aperture images are from the central view. However, using the 'linspace' function to create the vector showing the amount by which each sub-aperture image needs to move to refocus at a particular depth, creating errors in the generated focal stack images and causing the depth map to be inaccurate. Table 5.2 shows the values we use to shift the sub-aperture images for refocusing. The values in Table 5.2 are calculated by simply multiplying the position value of the sub-aperture view from the central view to the slope at which we must focus for the focal stack image. Using the values in Table 5.2, we could remove the error with the depth labels in the depth map.

The other issue we faced with the algorithm was that we could not increase the depth

TABLE 5.2: Light field refocusing shift variable from our approach

Slope	Sub-aperture image distance from central view								
	4	3	2	1	0	1	2	3	4
	Number of pixels by which the sub-aperture images are shifted								
0	0	0	0	0	0	0	0	0	0
1	-4	-3	-2	-1	0	1	2	3	4
2	-8	-6	-4	-2	0	2	4	6	8
3	-12	-9	-6	-3	0	3	6	9	12
4	-16	-12	-8	-4	0	4	8	12	16

map precision by increasing the number of focal stack images. This is because we were using gridded interpolation to shift the sub-aperture images. The gridded interpolation algorithm could not move the sub-aperture images with sub-pixel accuracy, meaning that if we tried to generate the focal stack with less depth difference between consecutive focal stack images, the two images were indistinguishable. This affected the accuracy of the refocused images in the focal stack, so we could not increase the depth map precision by increasing the number of focal stack images using gridded interpolation. On the other hand, using the frequency domain approach to shift the sub-aperture images, we can move the sub-aperture images with sub-pixel accuracy, which allows us to increase the depth precision of the estimated depth maps. Using the frequency domain approach to shift the sub-aperture images, we reduced the difference in slope values between two consecutive images in the focal stack from 0.2 to 0.01. Since we could refocus images with sub-pixel accuracy, our depth estimation algorithm was able to distinguish the best pixel match with the central reference view, increasing the depth map precision.

In this approach, the relationship between the image shift in the spatial and frequency domains is shown in Eq. (5.3), where s_0 and t_0 is the slope value and u, v is the sub-aperture location. The amount by which the sup-aperture image has to be shifted to refocus at a particular depth is the product of s_0, u and t_0, v.

$$f(s+s_0, t+t_0) = F(u,v)e^{-j2\pi(\frac{us_0+vt_0}{N})} \tag{5.3}$$

The refocused images can then be generated by either averaging the shifted sub-aperture pixels or by using the median value of the pixels. Fig. 5.4 visually depicts the shifted sub-aperture images to achieve focus based on the selected slope value at which the image needs to be focused. The amount by which each sub-aperture image needs to shift is the product of the slope and the position of the sub-aperture image, as presented in Table 5.2. When we focus on the '0' slope, the sub-aperture images are shifted by zero pixels; hence, the refocused image is obtained by summing up the sub-aperture images as shown in Fig. 5.4 (A). When we focus the image closer to the camera(negative slope), the sub-aperture images move away from the central view, as shown in Fig. 5.4 (B). In contrast, if we focus on the focal plane's further end(positive slope), sub-aperture images move inwards toward the central view, as shown in Fig. 5.4 (C). It should be noted that the shift of the sub-aperture images shown in Fig. 5.4 is exaggerated for use of understanding. Fig. 5.5 shows a comparison of the blur around the depth discontinuities when focusing on the background for both the summing techniques. It is clear from the magnified parts shown in Fig. 5.5 (A), (B) and (C) that the amount of defocus blur around the depth discontinuities using the averaged pixel value is very large compared to the amount of defocus blur using the median value.

The technique used in this chapter estimates the depth map by matching image patches from the focal stack to the central sub-aperture image, thus it must be ensured that all the edges and textured regions in the image are well defined in both the central all-in-focus image as well as the focal stack images to minimize the number of misdetections. This problem is addressed by adding the gradient of the image to itself. The main advantage of adding the gradient relies on the fact that in refocused focal stack images, the textured regions in the image that are in focus maximally contribute to the gradient image, while the out-of-focus objects contribute the least. This pre-processing step ensures that the object boundaries and textured regions are exaggerated in the focal stack images drastically reducing the number of misdetected patches, reducing the dependence on the post-processing steps. As the purpose of the gradient addition step is only to enhance the textured regions and boundaries on the focal stack images that are in-focus, unless the shadows cause the region to become textureless in the image, our algorithm is not affected by this step. The comparison between the accuracy of the estimated depth map with and

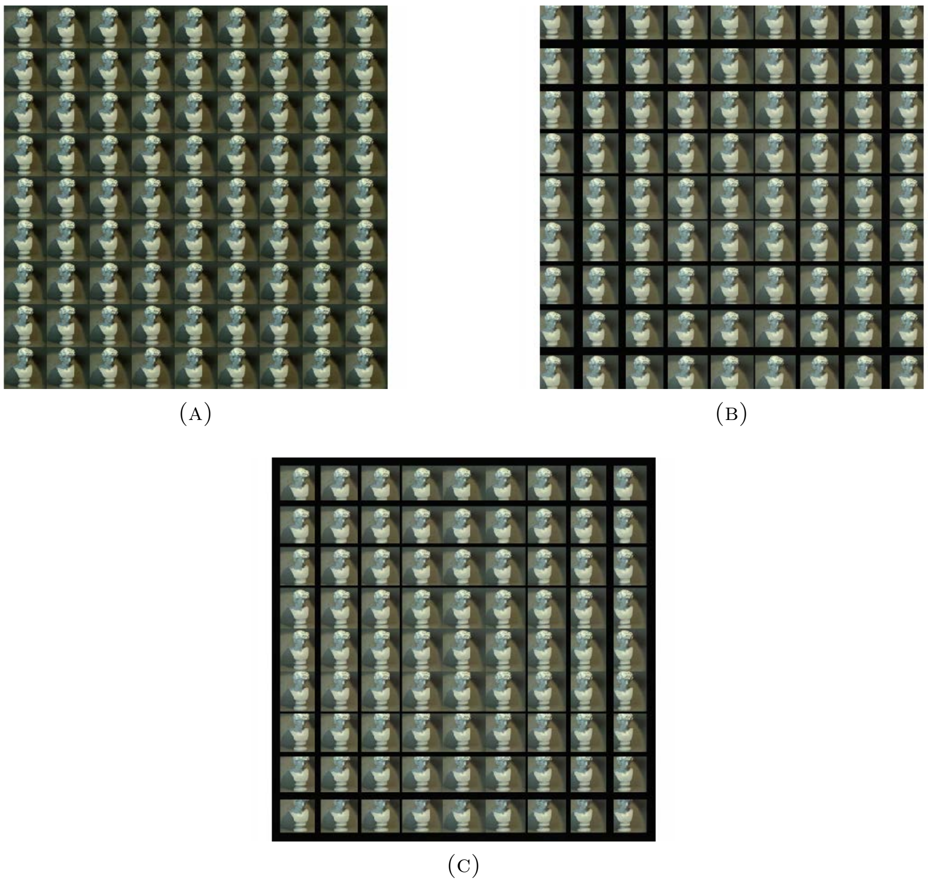


Figure 5.4: Shifted sub-aperture images to refocus using shift-sum filter for 'Antinous' LF image, (A) When refocus at '0' slope no shift is required, (B) When refocus for negative slope the images move away from the central view , and (C) When refocus for positive slope images towards the central view.

without the gradient addition is shown in Fig. 5.6 and it can be seen in Fig. 5.6 (C) and (F), that the part of the image with shadows are also estimated accurately.

5.2.3 Patch generation and comparison

The focal stack images acquired in the previous stage are divided into smaller image patches. Then those individual patches are compared with the corresponding block in the all-in-focus image. Since the approach's accuracy depends on the similarity check of the individual image patches, it is crucial that a block of appropriate shape and size is selected. Initial tests for patch selection were performed with a square patch of 10 x 10 pixels. This was the preliminary test that was performed to validate the approach. The test was

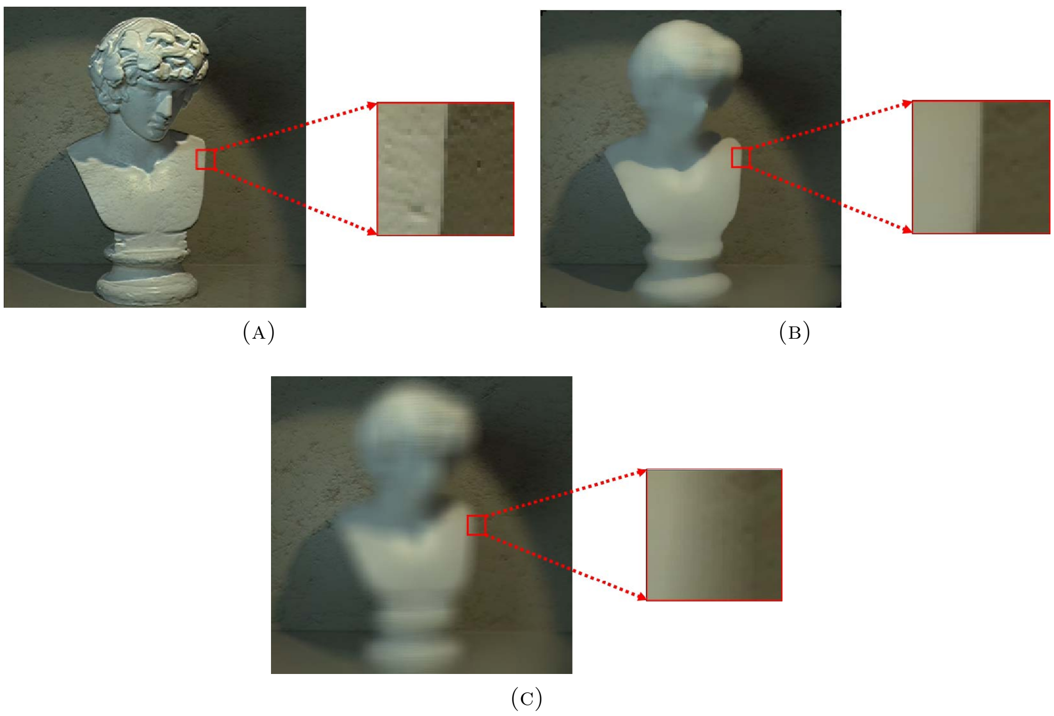


FIGURE 5.5: (A) Central sub-aperture view of the 'Antinous' LF image, (B) focal stack image refocused on the background using median pixel values, and (C) focal stack image refocused on the background using averaged pixel values and magnified region at depth discontinuity for each image.

then repeated with patches of different sizes. The results showed that smaller window sizes covered the image regions and boundaries more accurately. However, as the window size decreases, the processing time increases as the number of patches being compared increases. However, the nature of square-shaped patches over or under-compensated object boundaries that were slanted or curved in the image.

In testing cross-shape patches, the area that is uncovered in the gaps between four consecutive crosses is less than that of the primary cross window size. Cross patches of two different sizes were used to cover the entire image without any gaps. The primary cross's shape and size govern the secondary cross's shape and size, as shown in Fig. 5.7. An overlapping window is used to address misdetections due to the window size. We employ an overlapping window due to the limitations posed by a 4 x 4 patch size, which leaves only 2 x 2 pixels for the secondary window, indicated by the green square in Fig. 5.7. The reduction to 4 pixels for comparison significantly escalates misdetections. To enhance the accuracy of the depth map, equalizing the pixel count for both patches proves beneficial.

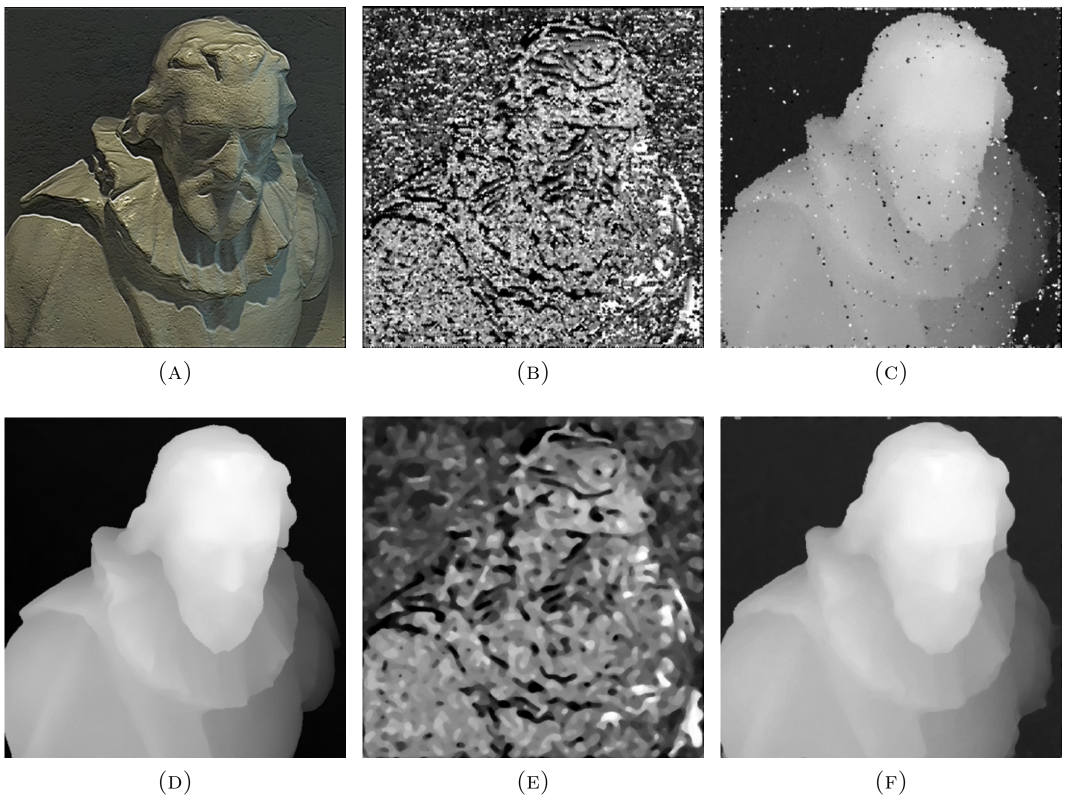


Figure 5.6: (A) Central sub-aperture view of the 'Cotton' LF image, (B) unrefined depth map calculated without gradient addition, (C) unrefined depth map calculated with gradient addition, (D) ground-truth depth map, (E) refined depth map calculated without gradient addition, (F) refined depth map calculated with gradient addition. The Badpix metric for without and with gradient addition is 9.11% and 96.23% respectively.

However, considering that the remaining pixels in the secondary patch are already factored in from the red square, we opt to utilize solely the 2 x 2 pixels from the green square during the depth map generation process. For the proposed algorithm, we use the two cross patches of size 4 x 4 pixels as shown in Fig. 5.7. In Fig. 5.7 the red and green squares are the pixels that are considered for matching with the all-in-focus image patch, but only the pixels highlighted in red in the red square and the pixels highlighted in green in the green square are used to generate the depth map. The patch size can be reduced to lower than 4 x 4 pixels. However, experimental tests revealed that using a patch smaller than 4 x 4 pixels does not improve the depth map accuracy and increases the computational time.

By comparing the FFT of the image patches, we are no longer looking at individual pixel values when comparing the image patches but a frequency domain representation of those patches, which makes the comparison more robust to noise. To illustrate the proposed

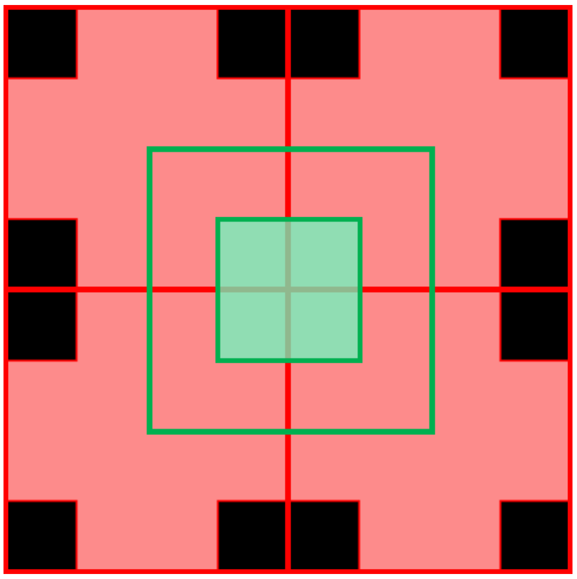


FIGURE 5.7: The red and green squares are the two overlapping 4 x 4 pixel patches used to cover the entire image. As the patches overlap, only the highlighted red and the green pixels from the red and green squares are used to estimate the depth.

approach, Fig. 5.8 shows the central sub-aperture image of the 'Dishes' LF image, a 4 x 4 pixel patch taken from the image and the FFT of the patch generated. The FFT shown in Fig. 5.8 (C) is a shifted version on the FFT: the highest co-efficient (DC) is in the centre. The focal length for synthetic images lies between a slope of -4 to +4 and for a Lytro camera, the image lies between slopes of -2 to +2. We, therefore correspondingly generated the focal stack from a slope of -4 to +4 for synthetic LF images and from the slope of -2 to +2 for real LF images. The slope interval between two consecutive focal stack images defines the number of depth levels in the depth map. We found through experimentation that for our work, the slope can be varied at an interval of 0.01, as using an interval below 0.01 does not show any significant change in the focus for consecutive focal stack images for both the synthetic and real LF images. Thus, for a synthetic LF image the depth map can have up to 801 depth levels as the focal stack is generated from a slope of -4 to +4 at a slope interval of 0.01, while for a real LF image the depth map can have up to 401 depth levels as the focal stack is generated from a slope of -2 to +2 at a slope interval of 0.01. The depth levels for the depth map can be increased by reducing the slope interval between the focal stack images. However, as the depth levels increase, the computation time also increases. For visual brevity in Fig. 5.9, only 8 refocused images are considered from -4 to +4 slope at an interval of 1. It is clearly seen that the fifth patch in Fig. 5.9 is

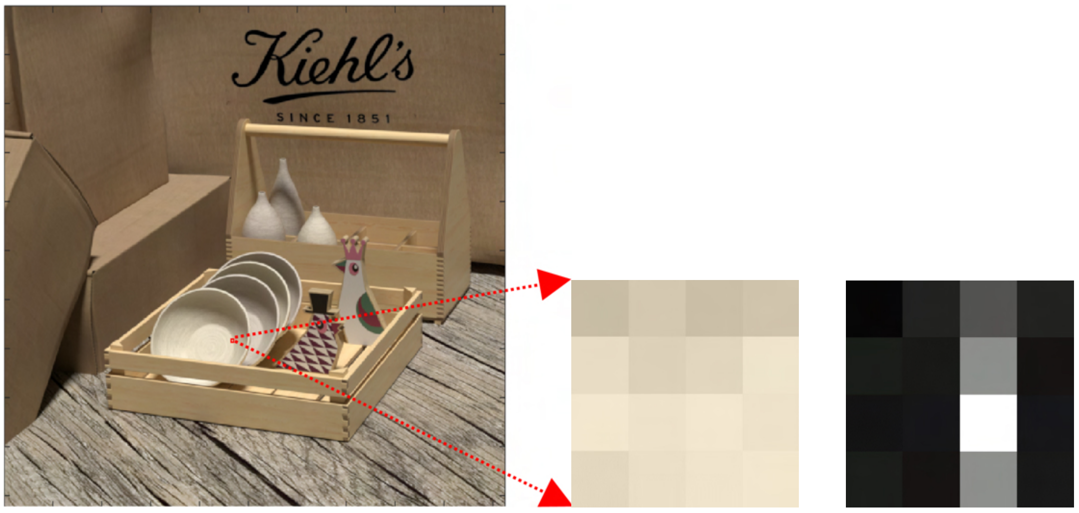
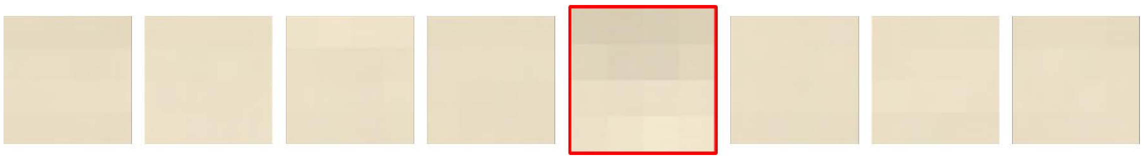


FIGURE 5.8: (A) Central sub-aperture image of the 'Dishes' LF image, (B) a magnified 6x6 RGB image patch, and (C) FFT of the image patch.

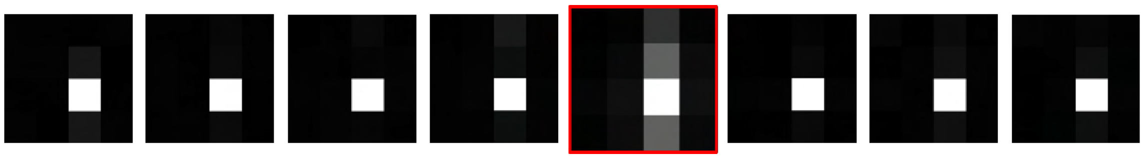
the most similar to the reference patch.

5.2.4 Depth map refinement

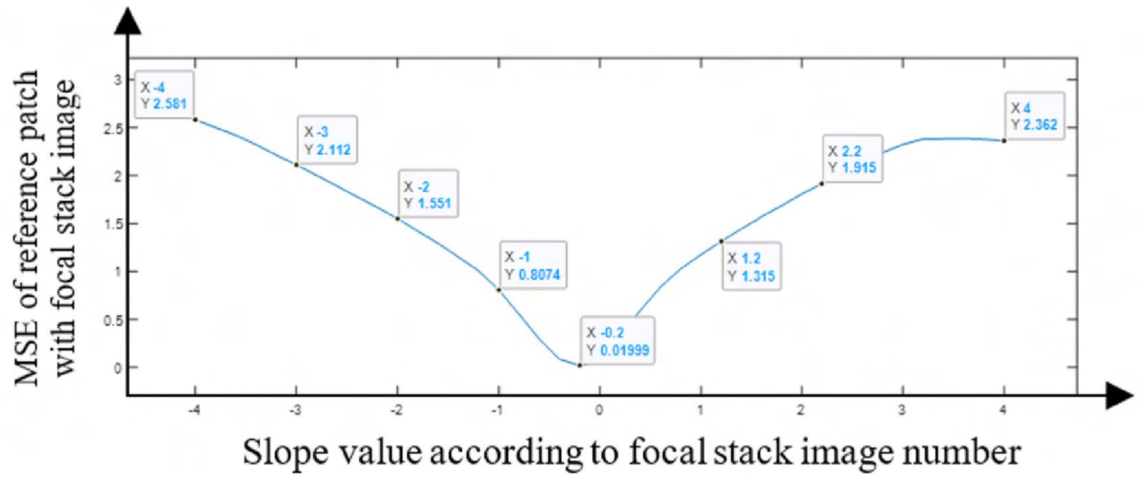
The depth map evaluated up to this stage has a few patches that are not detected correctly, and since the patches are shaped as a cross, it creates a depth map with jagged edges. Thus, the object boundaries need to be refined to restore the object's shape. Fig. 5.10 and Fig. 5.11 show the comparison between the ground truth and the estimated depth map before and after this refinement step for the synthetic and real images. The disparity map is refined in two steps using an iterative approach. Firstly, the histogram of the disparity map is checked for the number of pixels present at each depth. Suppose the number of pixels at a particular depth falls below the threshold value. We decided the threshold to be 100 pixels, which would mean that only six patches of 4 x 4 pixels in the entire depth map are at that particular depth, and we assume that as 100 pixels is such a small number, this is likely to result from misdetections. In that case, those pixels are filled with the maximum likelihood value of the pixels in the depth map at that position using the pixel value that occurs most in the neighboring pixels. The second step is similar to the first step, but the cross patches are considered instead of looking at individual pixels. This



(A)



(B)



(C)

FIGURE 5.9: (A) The RGB image patch and (B) the FFT patch at different focal lengths. The patch with the red boundary is the closet match to the reference patch in Fig. 5.8. (C) The graph shows the MSE values for the central image in Fig. 5.8, with the corresponding focal stack image patch.

step checks for isolated patches in the image with different surrounding depth patches. Once these patches are isolated, the patch is filled with the value of pixels with maximum likelihood in the depth map at that patch position using the pixel value that occurs most times in the neighboring pixels.

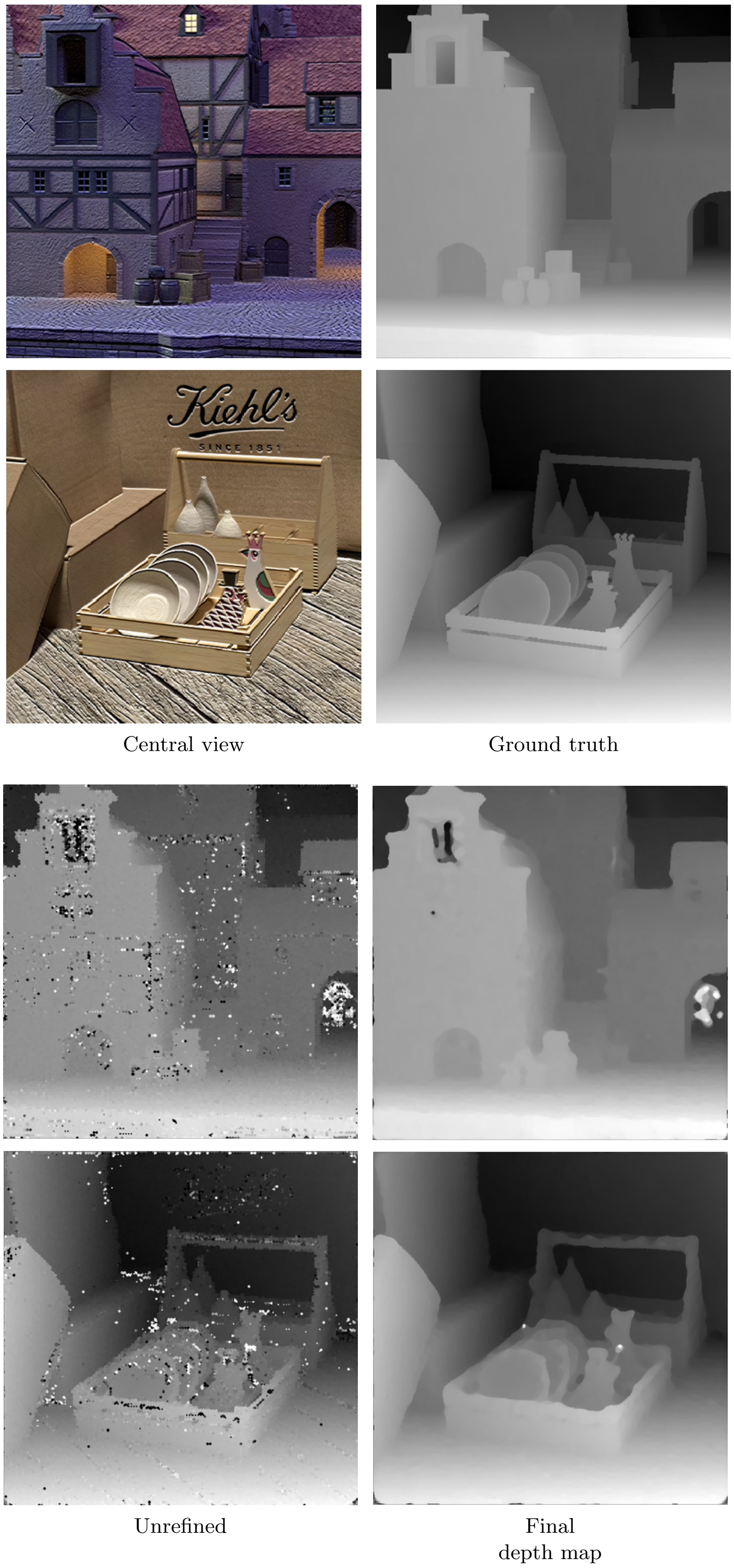


Figure 5.10: Comparison between the ground truth and the estimated depth map before and after the refinement step for synthetic images.

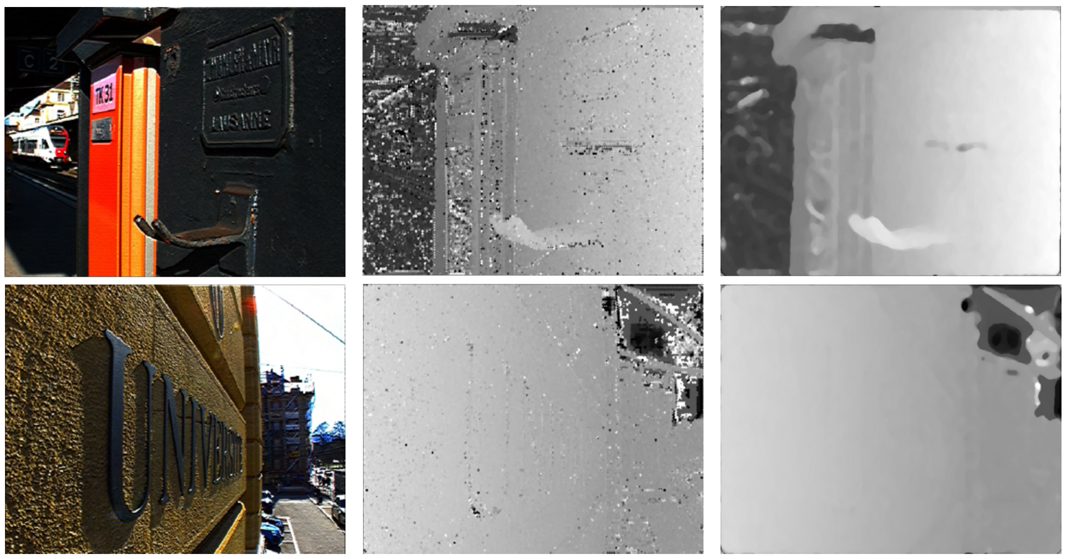


Figure 5.11: Comparison between the estimated depth map before and after the refinement step for real images.

5.3 Misdetection analysis

The depth estimation in the proposed algorithm compares the FFT of the all-in-focus patch and the patches at different focal positions in the focal stack to check for the least error to then accordingly estimate the depth. The quantitative results for 'Dot' image presented in Table 5.3 confirm that both techniques give comparable results. The metric "BadPix" [3] presented in Table 5.3 measures the percentage of pixels deviating by less than 0.07, 0.03, and 0.01 pixels from the ground truth in terms of their disparity values. This is currently the most common metric to validate depth map accuracy [3].

The advantage of using the FFT domain over the spatial RGB patch is that the number of misdetections is drastically reduced with the FFT, which reduces the algorithm dependence on the depth map refinement stage. The two main reasons for using the FFT domain over the spatial RGB patch are, firstly, as we are comparing the focal stack patches with the all-in-focus patches, most of the patches are out-of-focus. So, when comparing the RGB patches, the MSE for the out-of-focus patches can be closer to the MSE of the patch in-focus, causing misdetections. Figures 5.12 and 5.13 illustrate a notable phenomenon. In Fig. 5.12, the central sub-aperture image of the 'Antinous' LF image is displayed

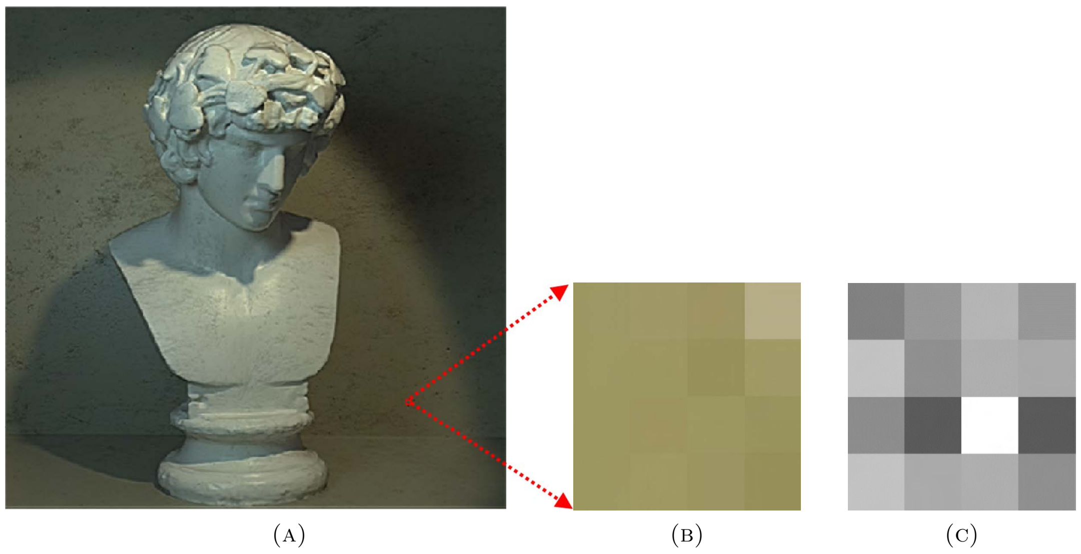
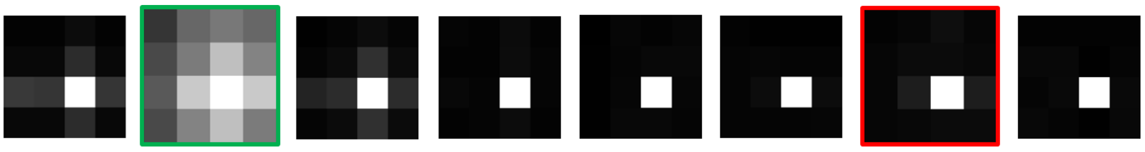


Figure 5.12: (A) Central sub-aperture image of the ‘Antinous’ LF image, (B) a magnified 4x4 RGB image patch, and (C) FFT of the image patch.

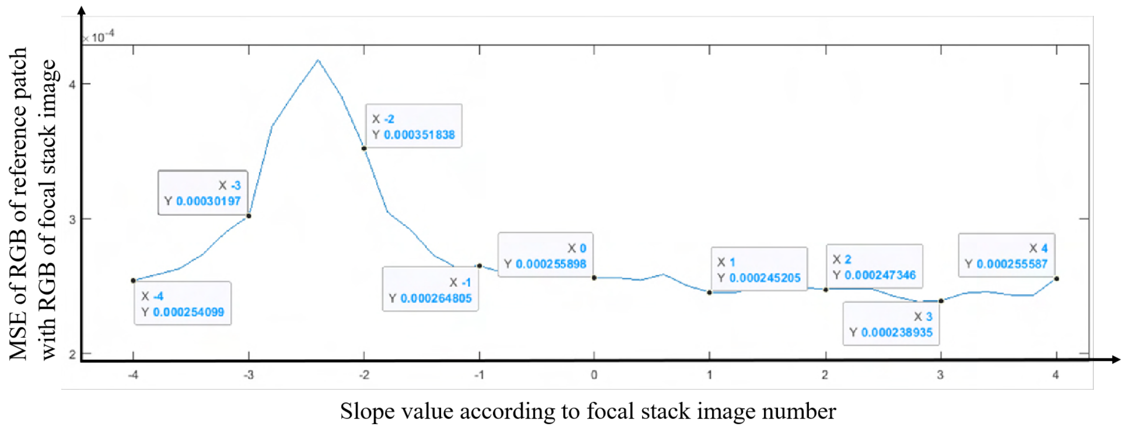
alongside a 4 x 4 pixel patch extracted from the image, along with its corresponding FFT representation. For conciseness in Fig. 5.13, only 8 refocused images are included, spanning from -4 to +4 slope at intervals of 1. As evident in Fig. 5.13 (A), the second patch closely resembles the reference patch depicted in Fig. 5.12 (B). However, when examining the Mean Squared Error (MSE) graph in Fig. 5.12 (C) comparing RGB patches, the seventh patch emerges as the most similar. Conversely, in the MSE graph displayed in Fig. 5.12 (D), comparing FFT patches, the graph exhibits a dip at the correct focal stack image patch.

The discrepancy in the RGB patch comparison arises from the fact that focal stack images do not precisely replicate the pixel values of the all-in-focus image. This variation in pixel values causes patches from out-of-focus focal stack images to closely match. In contrast, when comparing FFT patches, the out-of-focus patches lack frequency components, except for the color component due to blurring. This facilitates the accurate matching of the patches to the correct all-in-focus patch. Secondly, the experimental evaluation showed that FFT domain patches can better distinguish the correct depth when looking at patches closer to the patches in focus. The FFT domain patches are able to distinguish between patches taken from focal stacks at a very close slope value because the FFT of the patches also captures the variation in the brightness levels of the regions of the patches, making

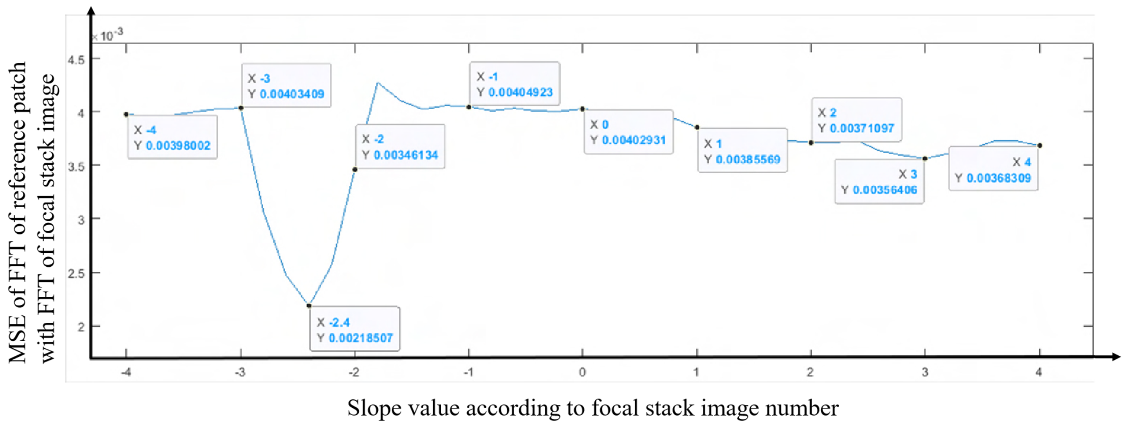
(A)



(B)



(C)



(D)

FIGURE 5.13: (A) The RGB image patch and (B) the FFT patch at different focal lengths. The patch with the green boundary is the closet match to the reference patch in Fig. 5.8, while the patch with the red boundary is misdetected in the RGB domain. (C) The graph shows the MSE values for the central image in Fig. 5.12, with the corresponding focal stack image patch for RGB patches. (D) The graph shows the MSE values for the central image in Fig. 5.12, with the corresponding focal stack image patch for FFT patches.

Table 5.3: Comparison of the estimated depth map when using the FFT patch and RGB patch algorithms

	Dots	Dots
	FFT patch depth map	RGB patch depth map
Badpix 0.07	**0.9705**	0.7760
Badpix 0.03	**0.8853**	0.5967
Badpix 0.01	**0.3880**	0.2391

it more robust for patches with less texture. A closer visual comparison in Fig. 5.14 also shows that the depth boundaries are sharper, and the depths are more accurately represented for the results using FFT. The 'dot' image is one of the more challenging images from the dataset due to the added Gaussian noise to approximate thermal and shot noise [3] and the shape and size of the objects in the image. In further evaluation, image-level comparisons for the proposed algorithm are shown in Fig. 5.14. It can be seen that the noise in the image considerably reduces the depth map accuracy when using RGB patches, where parts of objects in the image are completely misdetected. The results are more noise-resilient for the depth map generated by using the proposed FFT comparison. The proposed algorithm also outperforms the state-of-the-art for the 'dot' image, as shown in Fig. 5.15.

Fig. 5.16 shows the central view, the ground-truth depth map and the estimated depth map for the Rosemary image in the synthetic image dataset. Our algorithm produces an inaccurate depth map for the Rosemary image with a Badpix 0.07 value of 0.34. The error is caused because the wall in the background and the vase in the foreground have a smooth and textureless surface, which makes the two indistinguishable by our algorithm. It is important to note that the misdetection is not caused by the shadows in the image, as the carpet at the bottom of the image is not misdetected even though shadows fall on the carpet as well. The 'Cotton' LF image in Fig. 5.6 also shows that shadows do not affect the depth map accuracy for our algorithm.

Fig. 5.17 shows the Mean Squared Error (MSE) for the patch that is estimated to the correct depth at different focal lengths with the patch of the all-in-focus image. This trend in the graph indicates that the image patch is depicting the correct depth: in the example shown in Fig. 5.17, the plot takes a significant dip as it reaches the least MSE, which is the true depth for that particular patch. As the patch goes further away from the correct

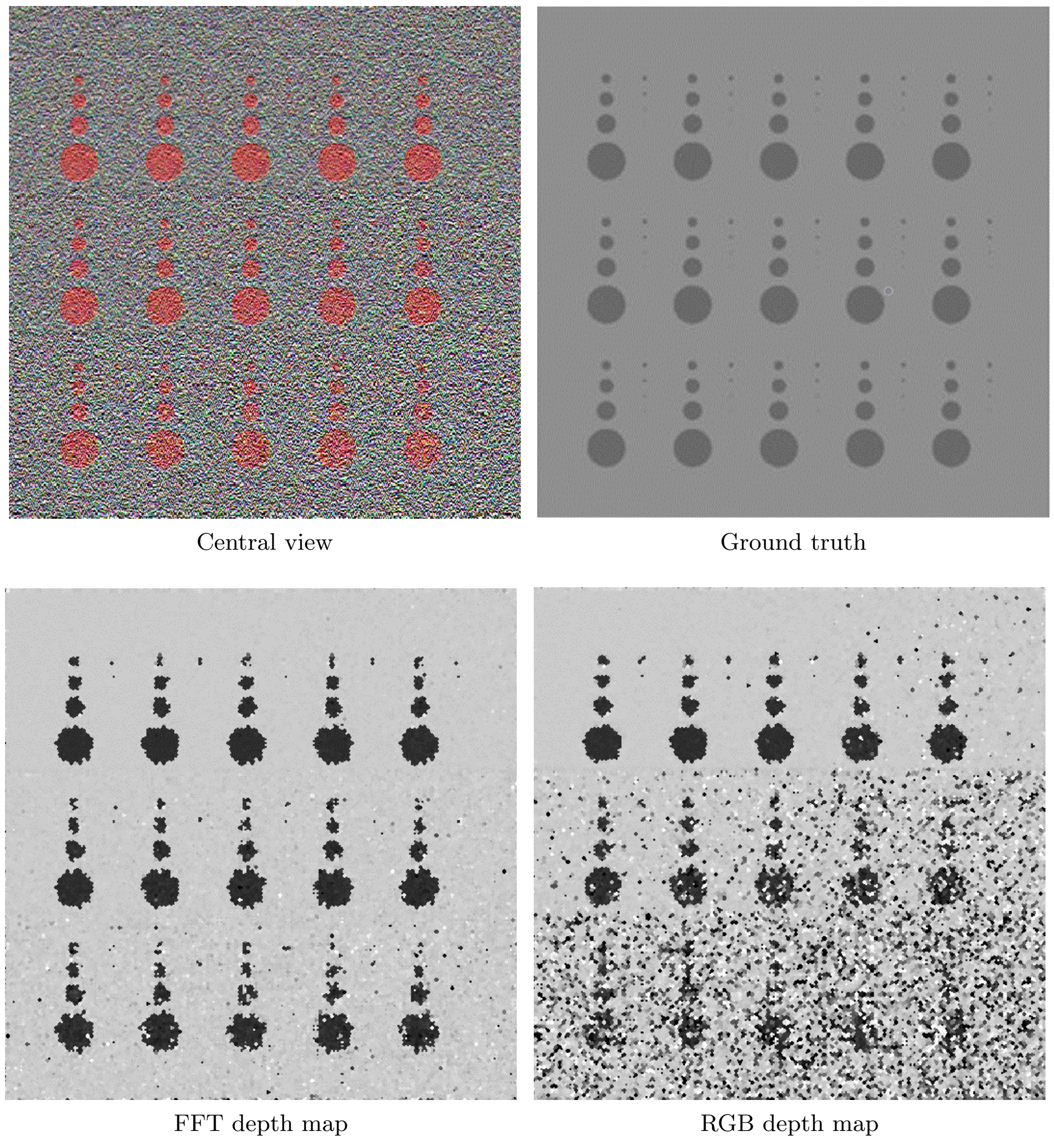


Figure 5.14: Visual comparison between the ground truth depth map, the result using FFT to estimate depth map (proposed algorithm) and depth map estimated using RGB patches.

depth value, the graph makes a similar curve on both sides of the focal stack. Even though in our work, we are only using the patches with the least MSE to estimate the depth map, it is important to see that the graph almost traces a bell curve. This shows that the MSE value is similar when defocusing toward or away from the patch in focus. In contrast, Fig. 5.18 shows an example of a patch being misdetected in the 'cotton' image. Although the graph follows a similar trend to Fig. 5.17, the graph has two considerable dips, one at the correct depth of slope 1.2 and the other at the incorrect depth of slope -2.6. It is also

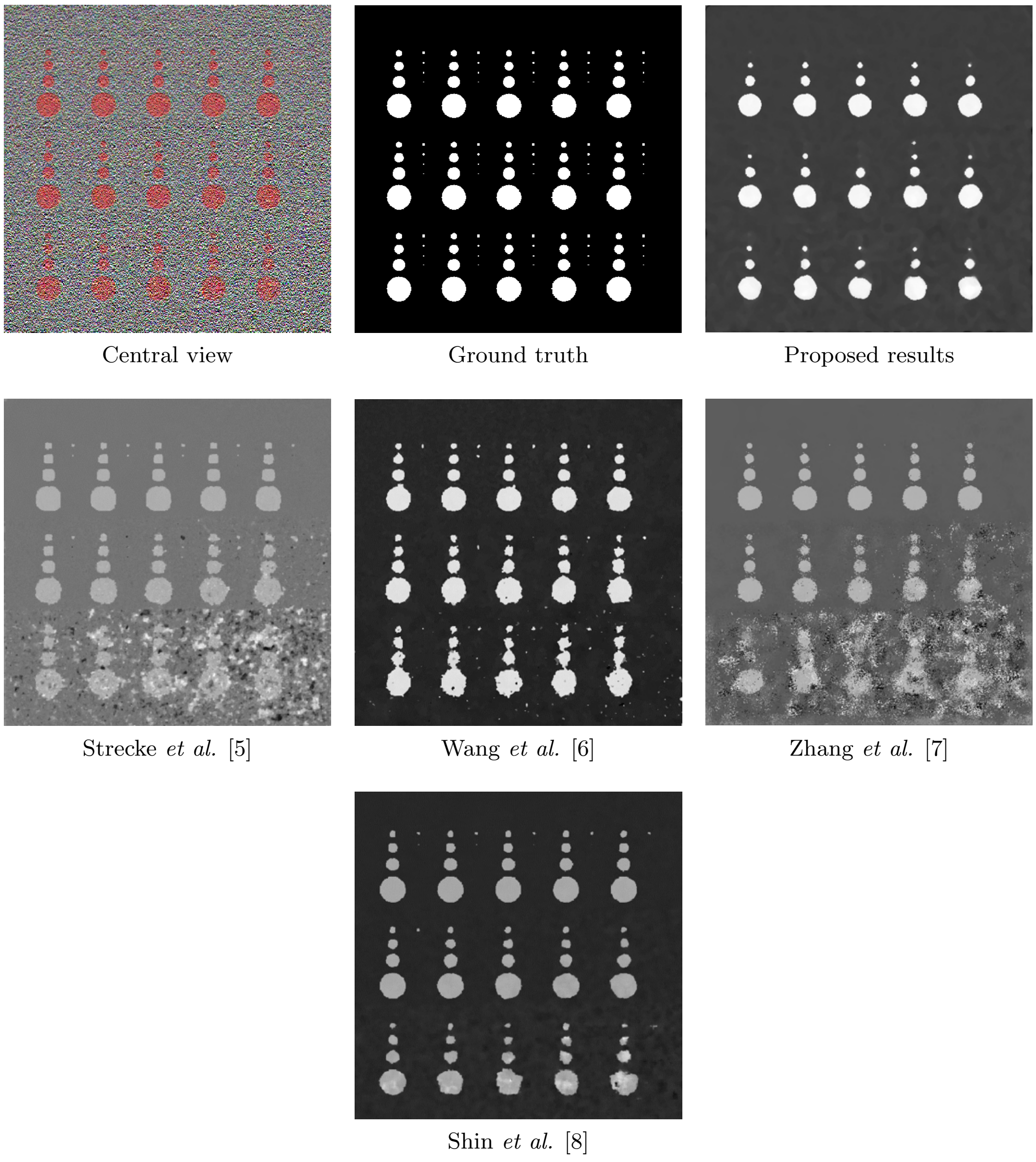


FIGURE 5.15: Visual comparison for the 'Dot' image with the ground truth, proposed and state-of-the-art algorithms.

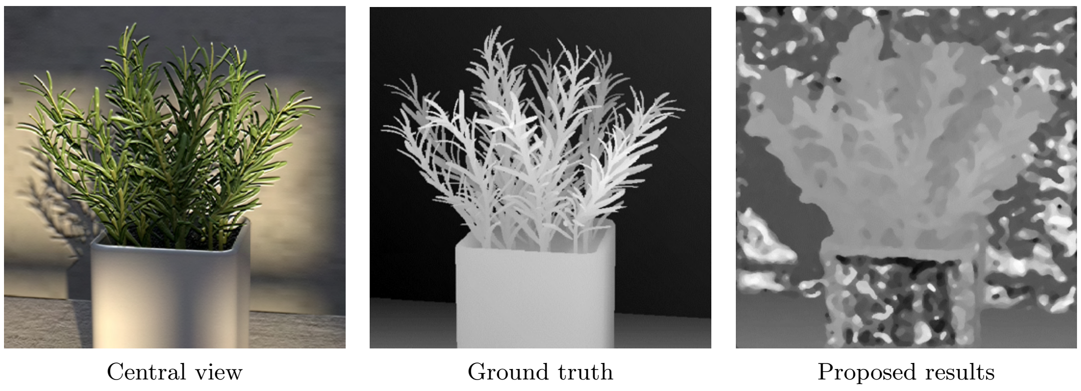


FIGURE 5.16: Misdetection of textureless regions in the 'Rosemary' image

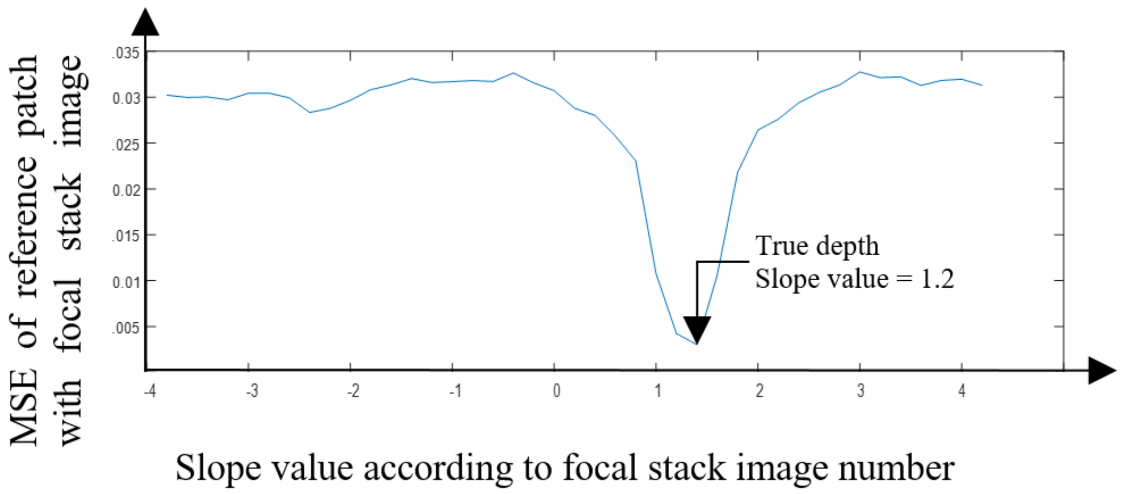
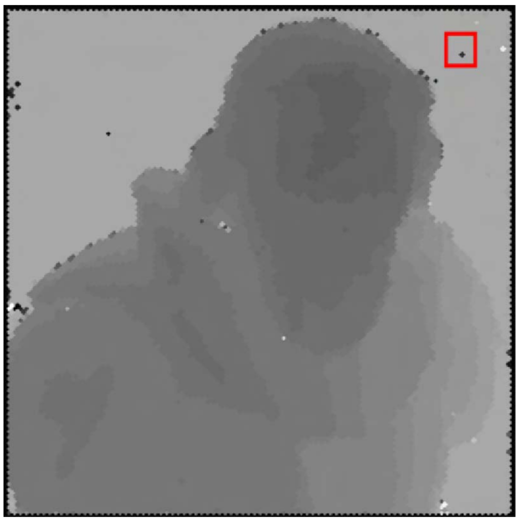


FIGURE 5.17: MSE for the central image patch and the patch at different focal lengths when the depth is estimated correctly.

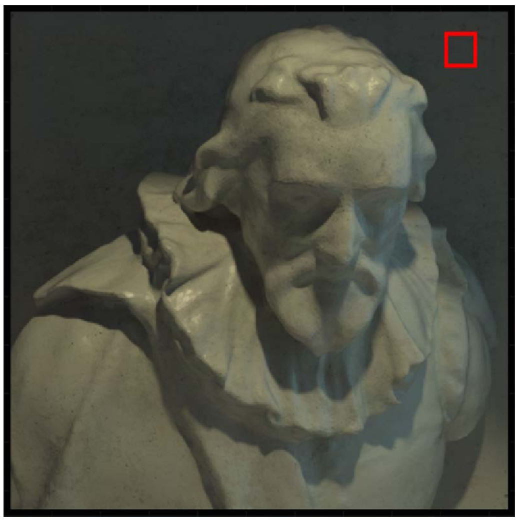
important to note that the number of misdetections is less, even though the depth map has not been refined at this stage.

5.4 Experimental results

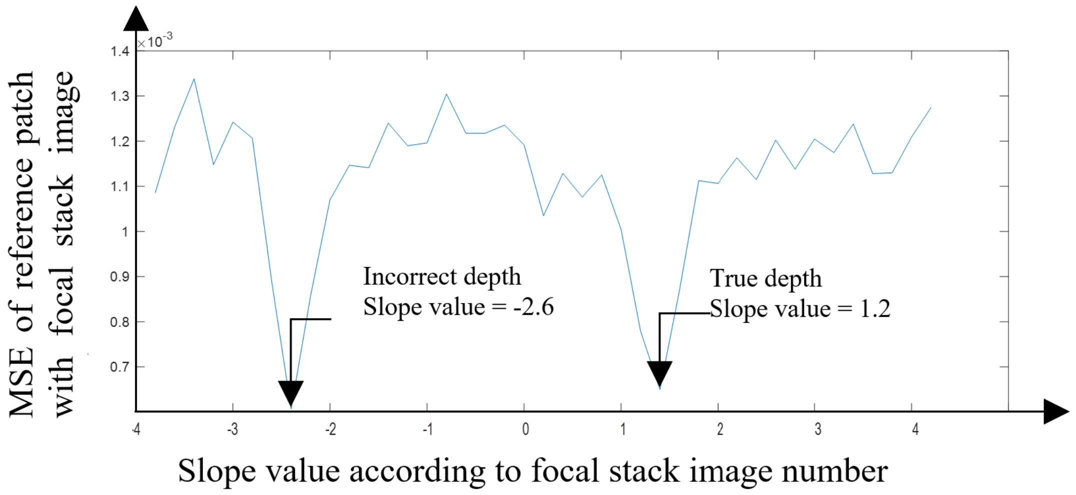
The results of the proposed algorithm were evaluated on both real and synthetic light field image datasets. The 4D light field dataset [3] was used for the synthetic data. The dataset is widely used to validate depth estimation algorithms for light field images as it contains ground-truth disparity and depth maps. The dataset contains 28 images with 9 x 9 sub-aperture images with a resolution of 512 x 512. We have selected ten images to



(A)



(B)



(C)

FIGURE 5.18: (A) The depth map of 'Cotton' image with the red square showing the cross error patch (B) The central image with the red box showing the error patch (C) The MSE of the central image patch with the patch at different focal lengths.

evaluate our algorithms with the benchmark algorithms, as each image contains different materials, lighting conditions and complex structures. Fig. 5.19 (A), (B), (C) and Fig. 5.20 (D) have finer detail and complex occlusions. Fig. 5.19 (C) and (D) have transparent and reflective surfaces. Fig. 5.19 (D), Fig. 5.20 (A) and (E) have shadows. Fig. 5.19 (B), Fig. 5.20 (B) and (C) are abstract scenes with complex textures. The EPFL light field dataset [94] was used for real data. The real image dataset contains 138 LF images in LFR (Light Field Raw) file format captured by a Lytro Illum camera with 15 x 15 sub-aperture images with a resolution of 434 x 625. The Lytro Illum camera used for LF image acquisition has different calibration data, and the LFR files were processed by their corresponding calibration data. Fig. 5.21 (A), (C), Fig. 5.22 (A), (B) and (C) contain finer objects and complex surfaces like perforated metal and fences. Fig. 5.21 (B), Fig. 5.22 (C) and (D) contain textureless or overexposed regions like the sky. Fig. 5.21 (A), (B), Fig. 5.22 (D) and (E) show a gradually change in depth and Fig. 5.21 (E) contains complex structures like the branches and trees.

The depth maps generated by our proposed approach initially calculate the depth range. The depth levels vary according to each LF image's maximum and minimum depth values. The slope for real data is within the range of +2 to -2, whereas for the synthetic data lies within the range of -4 to +4, and the slope interval used for both types of images is 0.01, as explained in section 5.2.2 and section 5.2.3. The number of depth levels can be increased or decreased by reducing or increasing the slope interval between focal stack images. The proposed algorithm is compared to four benchmark techniques from Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] using the BadPix metric that specifies the percentage of pixels where disparity deviates by less than 0.07, 0.03 and 0.01 pixels from the ground truth. We have chosen these four techniques as they are state-of-the-art for the different depth estimation techniques. Strecke *et al.* [5] and Wang *et al.* [6] use depth from defocus, Zhang *et al.* [7] use EPIs, whereas Shin *et al.* [8] use CNN for depth estimation. To assess the outcomes for all four techniques, we executed the code supplied by their respective authors in our evaluation.

In order to compare the depth maps using different algorithms to the ground truth, all output disparity maps are normalized to the ground truth depth map range. For Strecke *et al.* [5] and Shin *et al.* [8], normalized results are directly compared to the ground-truth disparity map. For Wang *et al.* [6] and Zhang *et al.* [7] the disparity map is normalized before comparing it to the ground-truth.

5.4.1 Synthetic LF images

The LF images in the 4D Light Field Dataset [3] comprise 9 x 9 sub-aperture images. For the synthetic images, the images in Fig. 5.19, Fig. 5.20, Fig. 5.24 and Fig. 5.25

show the error pixels in red where depth deviates by more than 0.07 from the ground truth and the pixels in green where depth deviates by less than 0.07. Table 5.4 and Table 5.5 compares the ground truth images to the disparity maps generated by the algorithms that are being tested using the Badpix metric. The LF images generated synthetically have little to no noise compared to the real LF images. Thus the estimated depth maps have fewer misdetections, and the depth boundaries are well defined on the synthetic data compared to the real data. Table 5.4 and Table 5.5, which is a comparison of the depth maps with the ground-truth depth map, shows that the proposed algorithm outperforms the state-of-the-art algorithms in the two criteria of Badpix 0.07 and 0.03 for the 'dots' images from the synthetic images in the dataset. On visual inspection of Fig. 5.19 and Fig. 5.20, we can observe that even though the noise level is increased in the bottom part of the image, the background is region is still detected accurately.

For the 'medieval 2' image, the region in the image near the window on the top left and near the door on the bottom right has a dark shadow which is a common area misdetected for all algorithms. Shin *et al.* [8]'s produce the least errors around object boundaries for the synthetic light field images. The 'kitchen' and 'museum' image in Fig. 5.19(C) and (E) shows how the error pixels are in the same regions for all the estimated depth maps. The similarity is that those regions in the image are either transparent or reflective surfaces. And Shin *et al.* [8] show lesser errors around these regions as they explicitly use a mask for these types of surfaces while training their network. The depth map for our algorithm, Shin *et al.* [8] and Strecke *et al.*[5] give similar results for the background and foreground region in the 'kitchen' and 'museum' image, whereas the depth maps from Wang *et al.* [6] and Zhang *et al.* [7] produce errors in the background and foreground regions.

For the 'museum' image in Fig. 5.19(E), and the 'pillow' and 'platonic' images in Fig. 5.20(A) and (B), our proposed algorithm out-performs the non-CNN based algorithms at Badpix 0.03, with comparable results for the other two criteria. Out of the three images mentioned above, the main reason for the errors is the reflective display case and the bright display case lighting for the 'museum' image. For the 'platonic', 'pyramids' and 'tomb' images in Fig. 5.20, our depth map generates errors only at depth boundaries and all other regions are estimated accurately and is comparable to Shin *et al.* [8]'s CNN approach. Shin *et al.* [8] produce high accuracy depth maps and can also distinguish accurate depths for

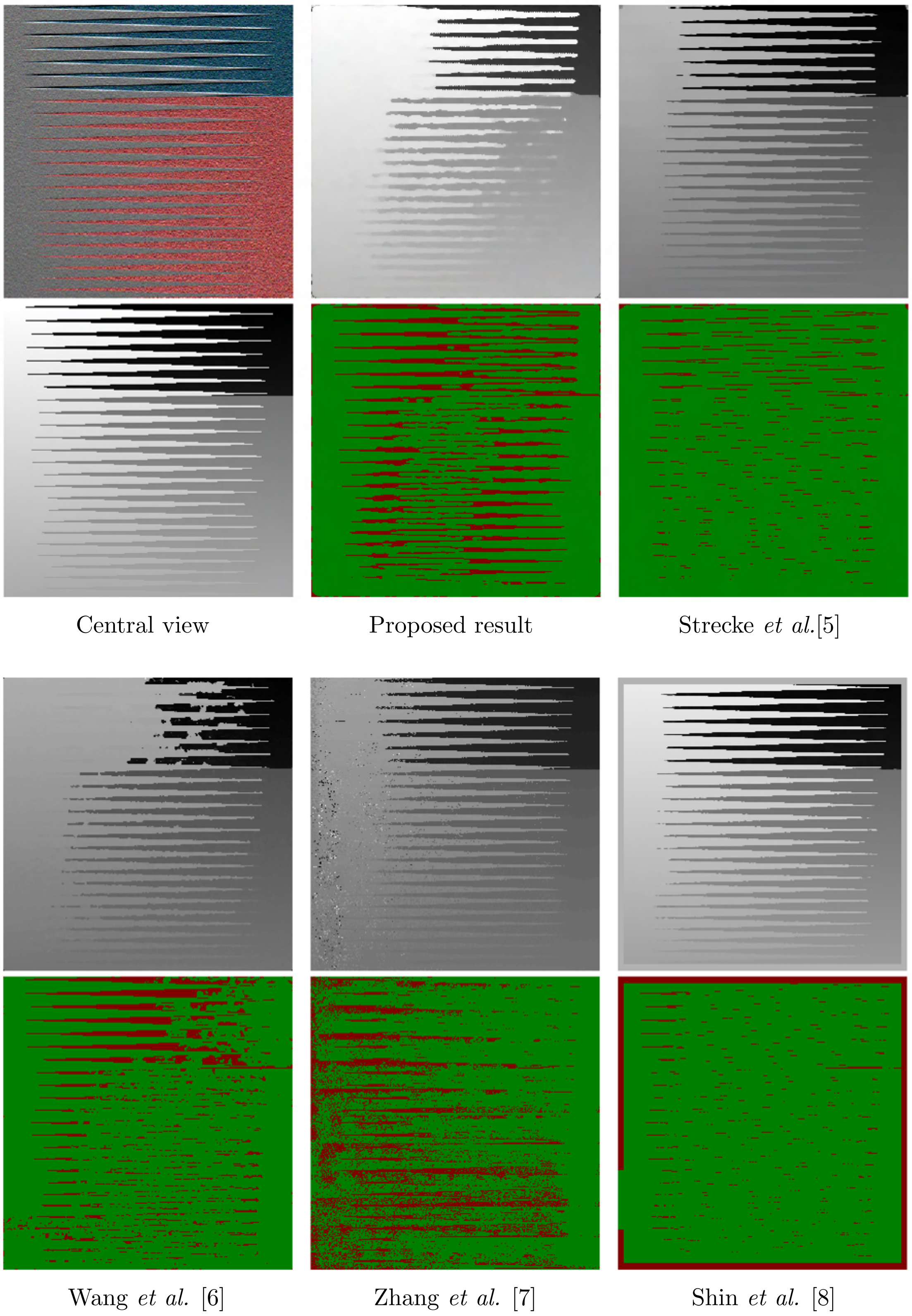
TABLE 5.4: Quantitative depth map comparison to ground truth for synthetic data

	Back-gammon	Dots	Kitchen	Medi-eval2	Museum
			Proposed Results		
Badpix7	0.8230	**0.9605**	0.7010	0.9362	0.8440
Badpix3	0.7324	**0.8853**	0.5941	0.8528	0.7772
Badpix1	0.4910	0.3880	0.3749	0.5514	0.5305
			Strecke *et al.* [5]		
Badpix7	0.9580	0.6273	0.7224	0.9608	0.8578
Badpix3	0.9283	0.4514	0.6282	0.8895	0.7615
Badpix1	0.6606	0.1777	0.4644	0.6469	0.5256
			Wang *et al.* [6]		
Badpix7	0.8753	0.8801	0.6300	0.5136	0.8522
Badpix3	0.4525	0.2485	0.3991	0.1119	0.6902
Badpix1	0.0544	0.0456	0.1772	0.0370	0.2741
			Zhang *et al.* [7]		
Badpix7	0.7889	0.7358	0.6379	0.9580	0.8940
Badpix3	0.3762	0.4810	0.3165	0.7513	0.5413
Badpix1	0.1057	0.4810	0.0997	0.2658	0.1899
			Shin *et al.* [8]		
Badpix7	0.9777	0.9473	0.7931	0.9847	0.9598
Badpix3	0.9594	0.7957	0.7209	0.9584	0.9053
Badpix1	0.8265	0.5122	0.4809	0.7263	0.6478

occlusions and smaller objects in the image. Still, the accuracy is reduced for transparent or reflective surfaces and noisy images.

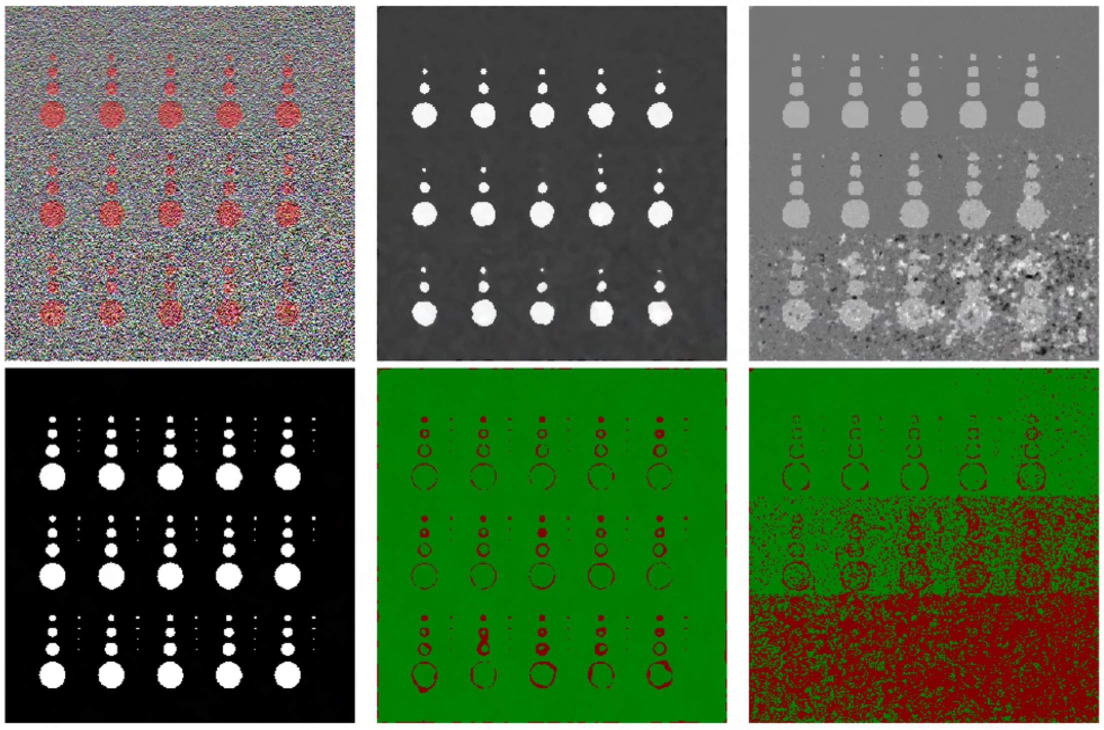
5.4.2 Real LF images

The proposed algorithm is not able to distinguish objects in the image that are less than 4 x 4 pixels in width due to the patch size used, but using patches of size less than 4 x 4 pixels drastically increases the number of misdetected depth patches and also increases the computational time. The image results displayed in Fig. 5.21 and Fig. 5.22 on visual inspection shows similar outcomes as for the synthetic images. The central view for four images in the Fig. 5.21 (A), (B), and Fig. 5.22 (D) and (E) show a gradual change in depth and the depth maps correspondingly show the gradient change. For the proposed algorithm, the images in Fig. 5.21 (A), (C), and Fig. 5.22 (C) with the chain fences, the regions where the chain has a shadow cast over it is mis-detected. The chain fences in all three images for all the algorithms have been under or over-compensated. The lorikeet

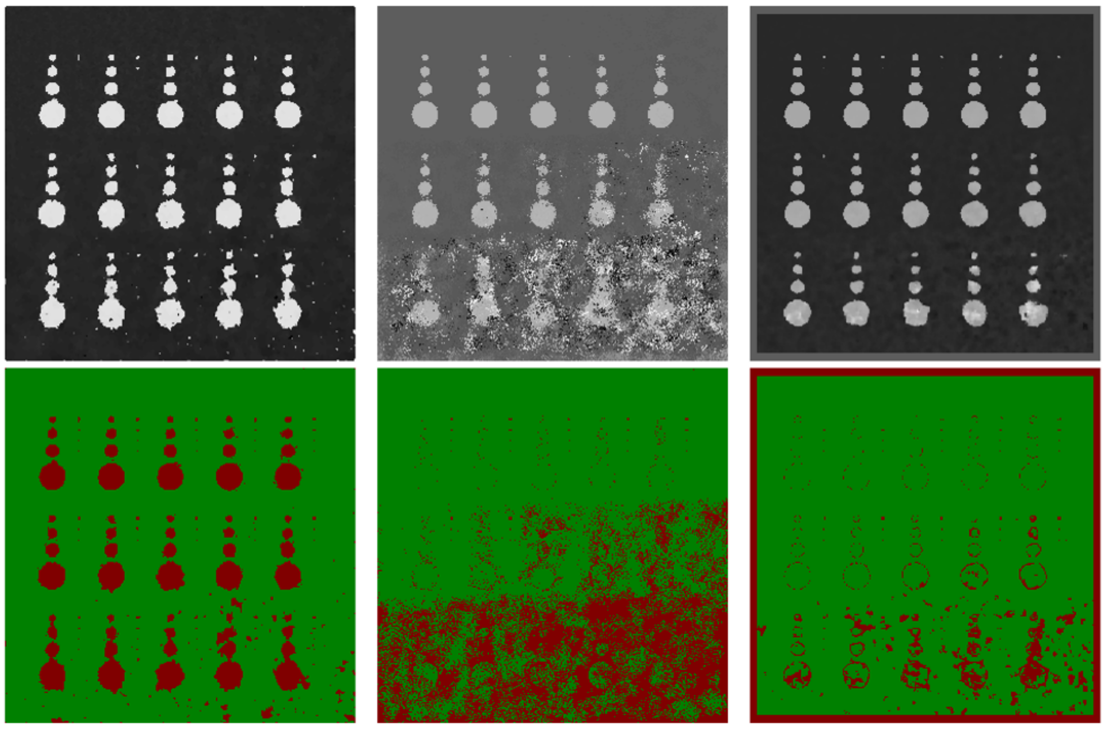


(A) 'Backgammon' image

Figure 5.19: *Cont.*



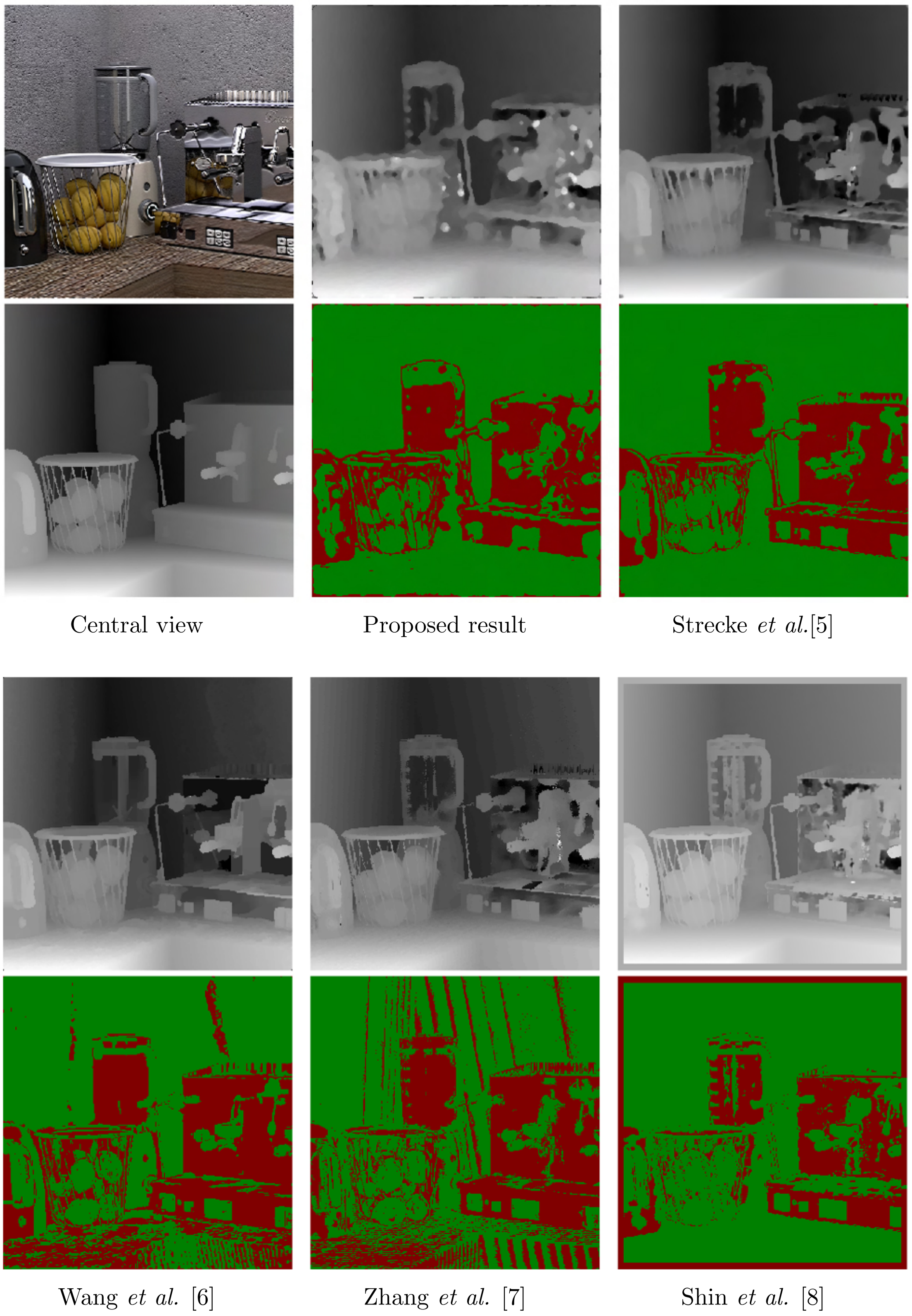
Central view Proposed result Strecke *et al.*[5]



Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

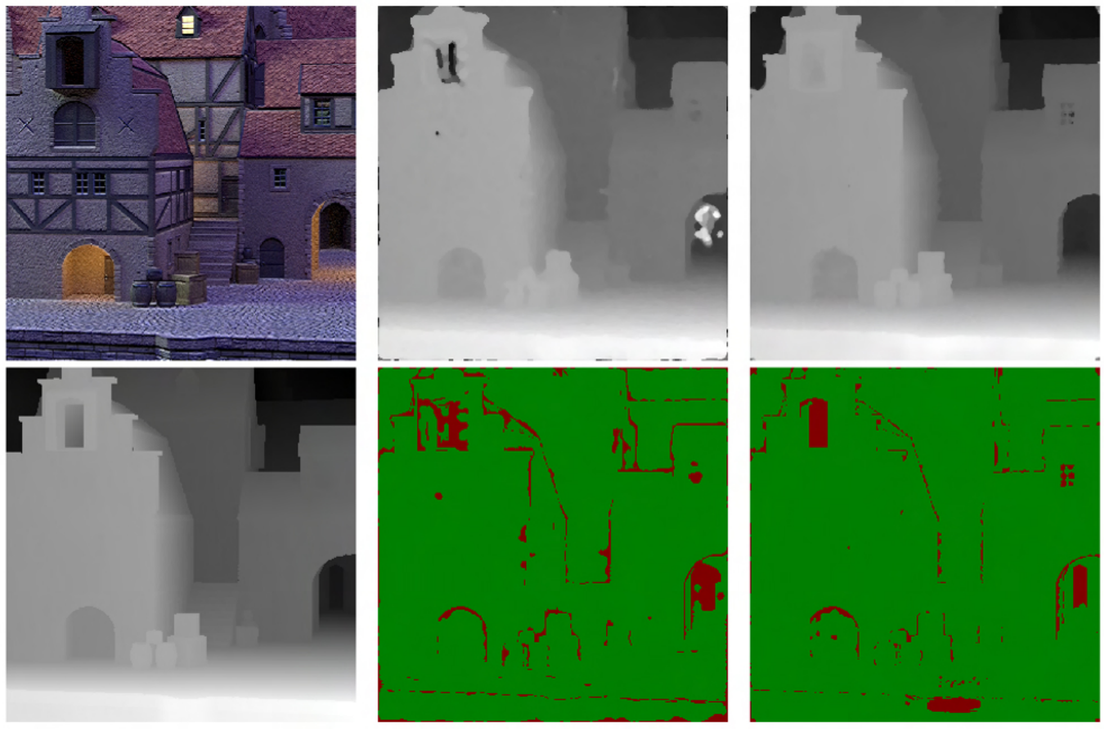
(B) ‘Dots’ image

FIGURE 5.19: *Cont.*

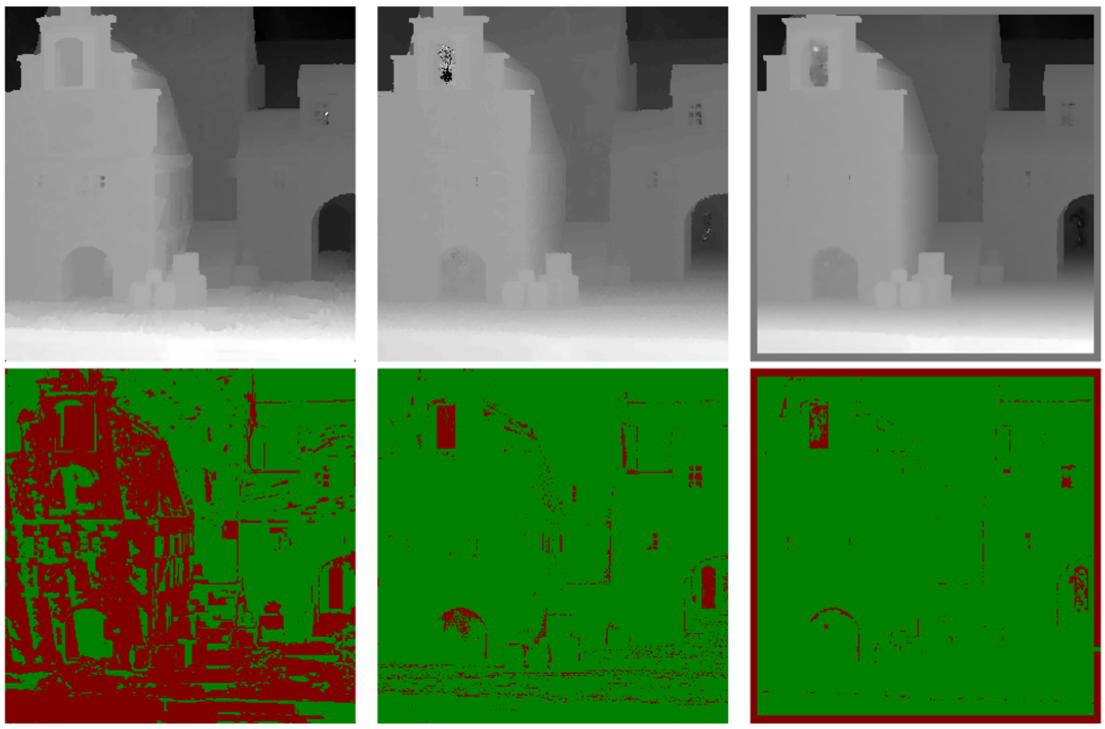


(C) ‘Kitchen’ image

FIGURE 5.19: *Cont.*



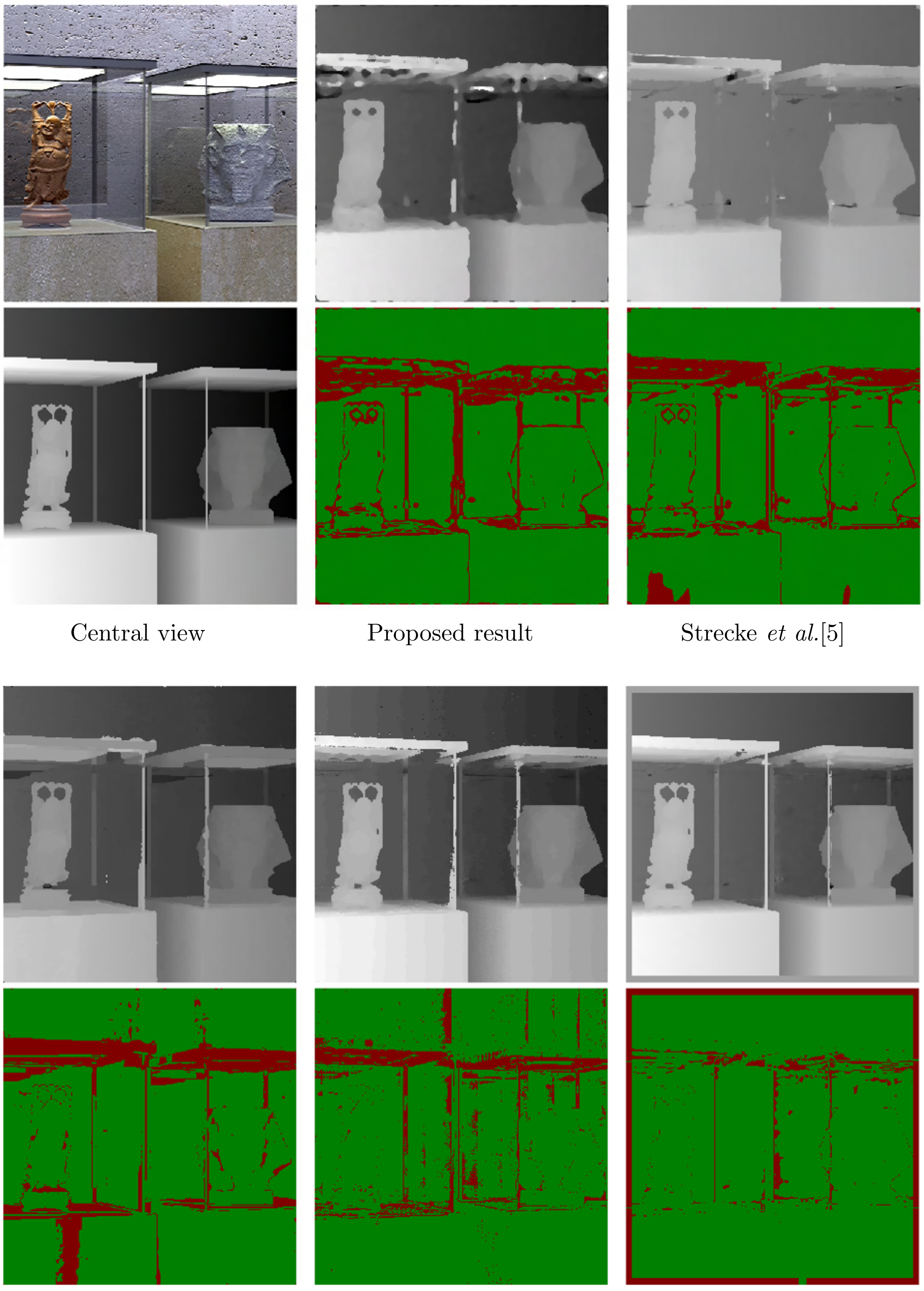
Central view Proposed result Strecke *et al.*[5]



Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

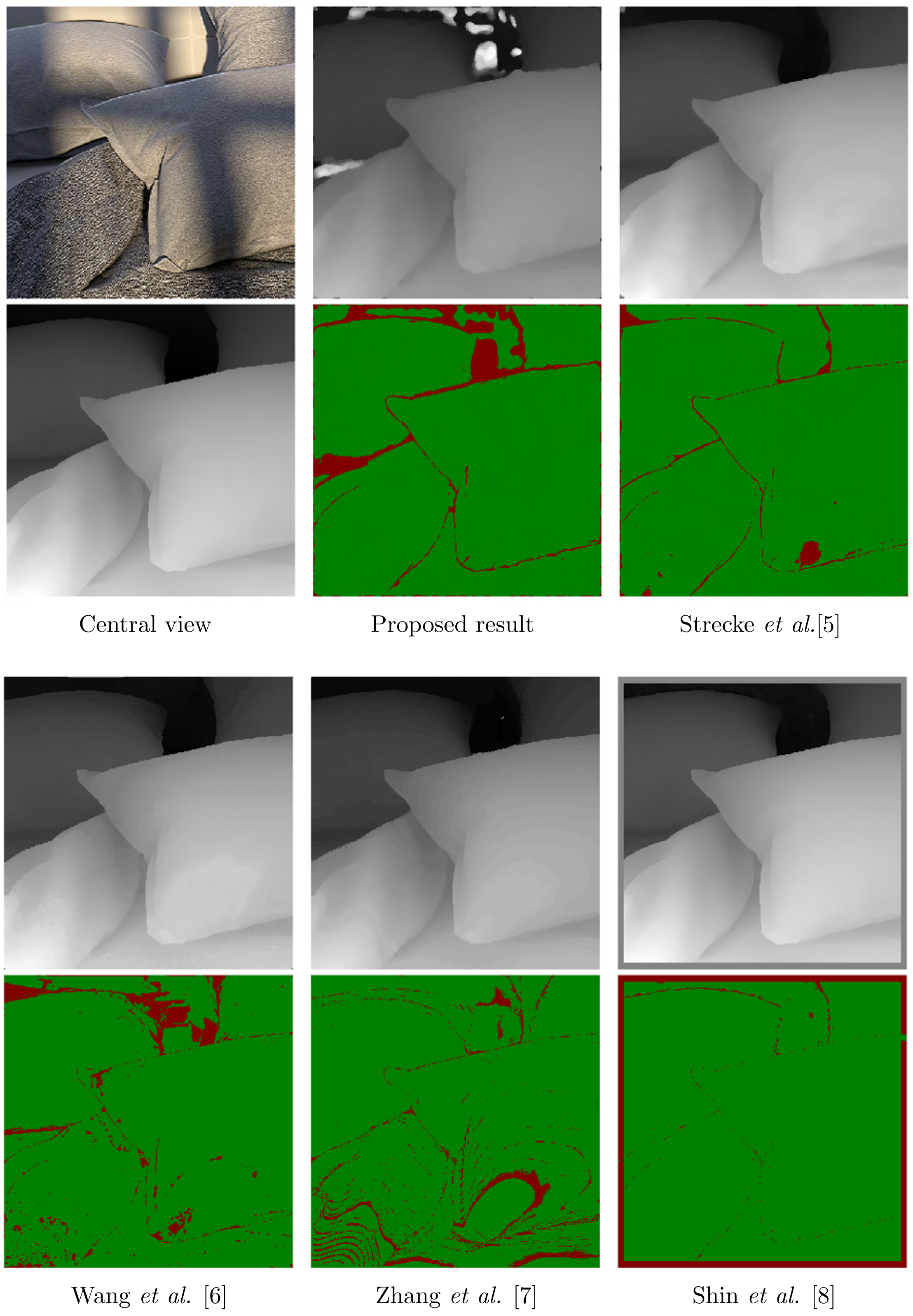
(D) 'Medieval 2' image

FIGURE 5.19: *Cont.*



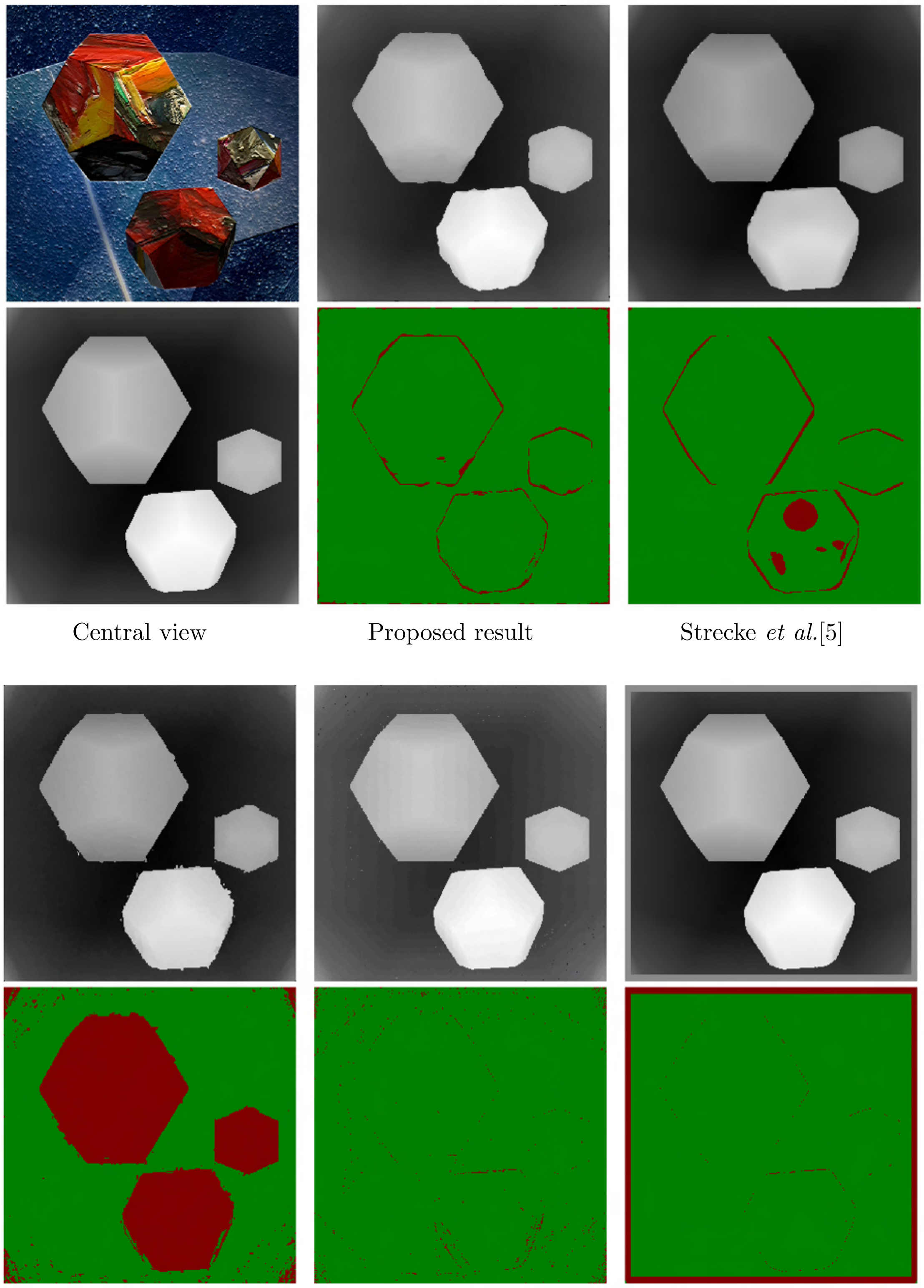
(E) ‘Museum’ image

FIGURE 5.19: Visual comparison of the proposed algorithm with Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for synthetic LF images.



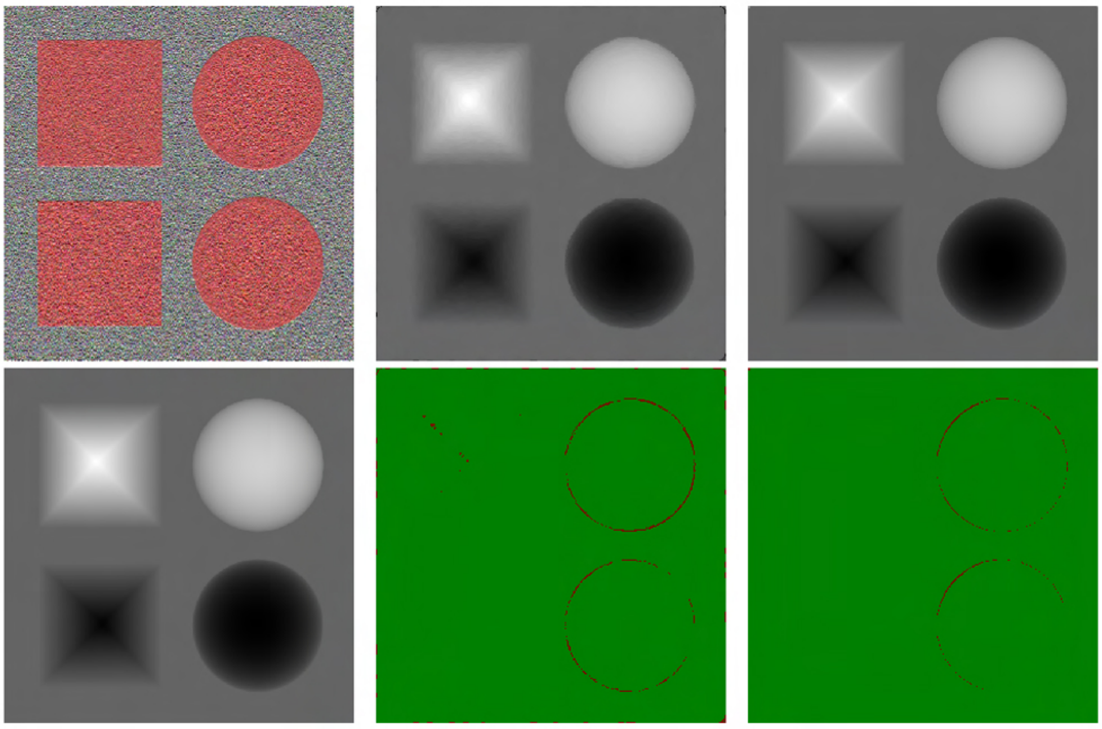
(A) ‘Pillows’ image

Figure 5.20: *Cont.*

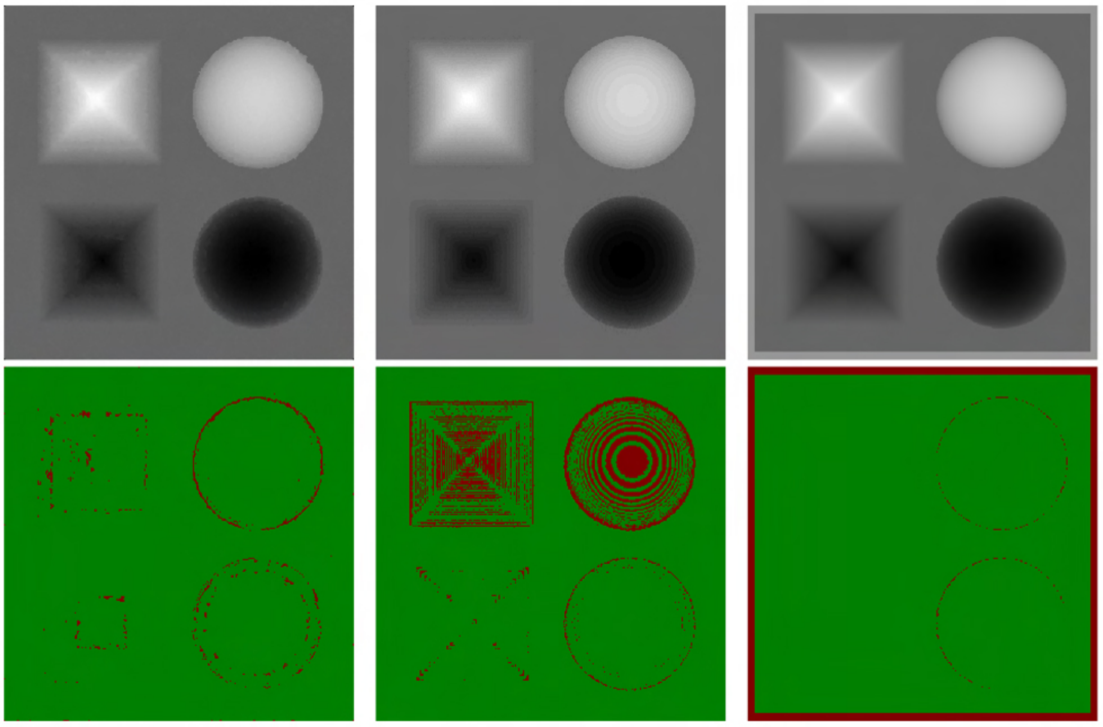


(B) 'Platonic' image

FIGURE 5.20: *Cont.*



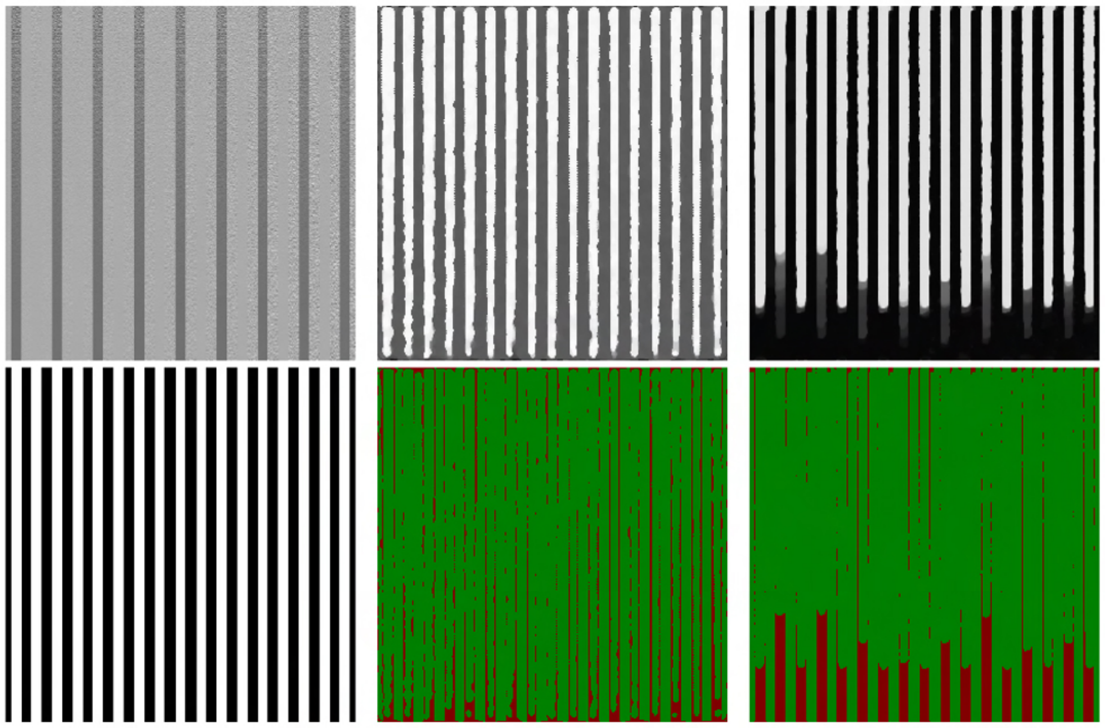
Central view Proposed result Strecke *et al.*[5]



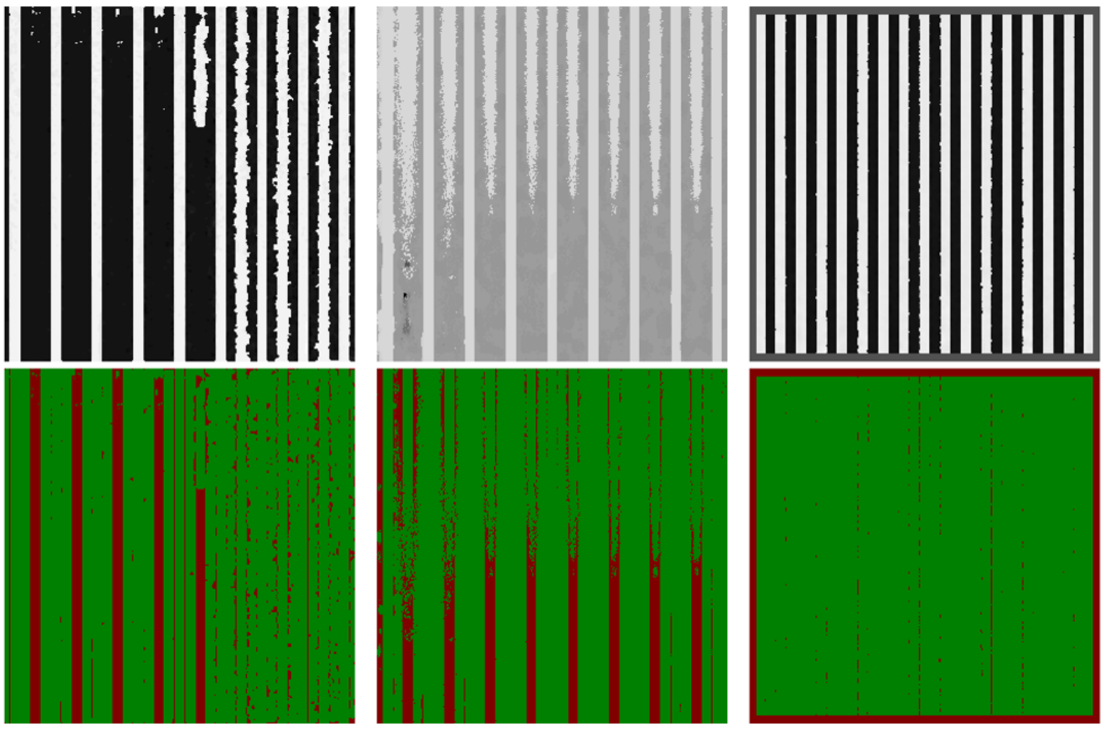
Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(C) ‘Pyramids’ image

Figure 5.20: *Cont.*



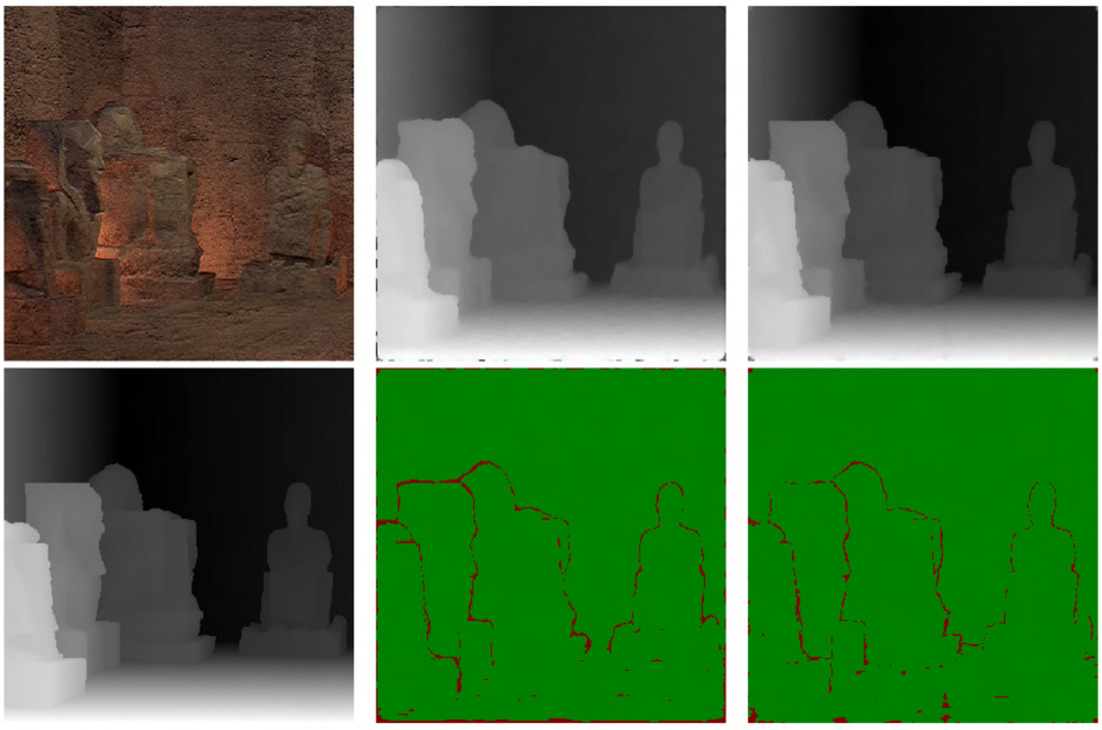
Central view Proposed result Strecke *et al.*[5]



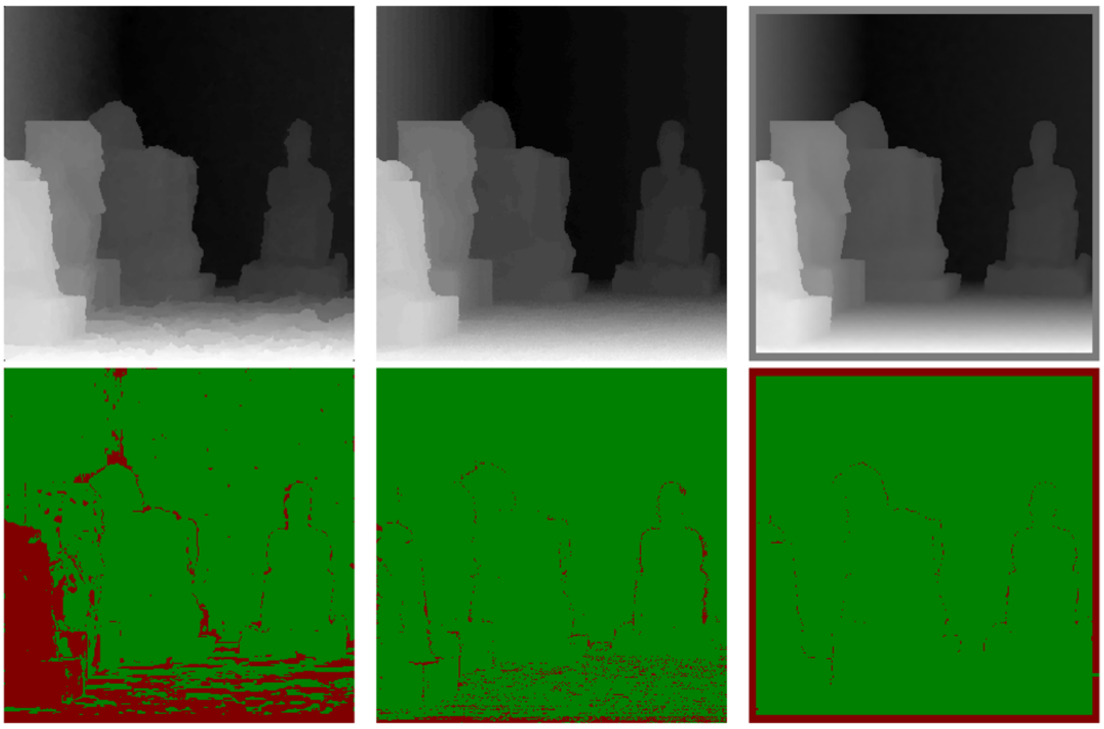
Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(D) ‘Stripes’ image

FIGURE 5.20: *Cont.*



Central view Proposed result Strecke *et al.*[5]



Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(E) ‘Tomb’ image

FIGURE 5.20: Visual comparison of the proposed algorithm with Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for synthetic LF images.

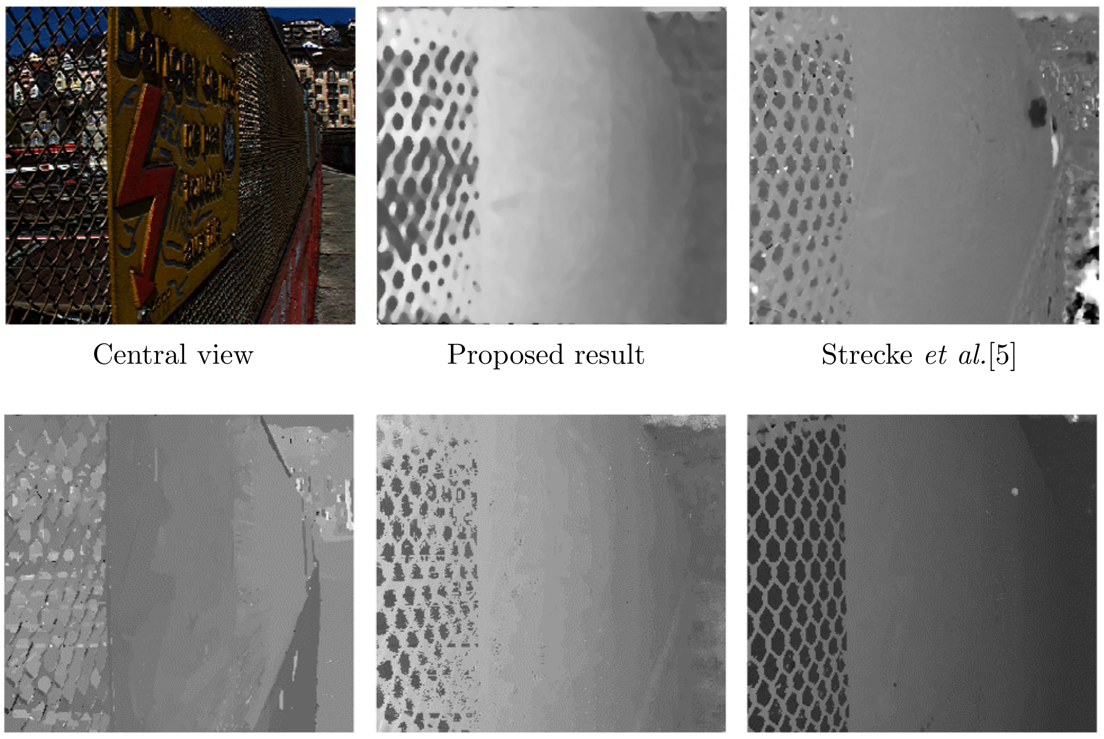
Table 5.5: Quantitative Depth Map Comparison for Synthetic Data to Ground Truth for different Algorithms

	Pillows	Platonic	Pyramids	Stripes	Tomb
		Proposed Results			
Badpix7	0.9212	0.9747	0.9920	0.8853	0.9696
Badpix3	0.8769	0.9447	0.9582	0.8275	0.9100
Badpix1	0.6096	**0.7600**	0.7485	0.6732	0.6423
		Strecke *et al.* [5]			
Badpix7	0.9710	0.9645	0.9969	0.8741	0.9813
Badpix3	0.8687	0.9230	0.9927	0.8556	0.9252
Badpix1	0.4914	0.7792	0.9417	0.4925	0.6875
		Wang *et al.* [6]			
Badpix7	0.9387	0.6583	0.9843	0.8231	0.7953
Badpix3	0.5611	0.4620	0.7520	0.0048	0.4134
Badpix1	0.1492	0.1889	0.0737	0.0004	0.1359
		Zhang *et al.* [7]			
Badpix7	0.9398	0.9906	0.8958	0.8373	0.9622
Badpix3	0.5066	0.7454	0.1885	0.5243	0.7500
Badpix1	0.1869	0.2946	0.0634	0.5243	0.2871
		Shin *et al.* [8]			
Badpix7	0.9939	0.9981	0.9972	0.9894	0.9963
Badpix3	0.9772	0.9941	0.9917	0.9865	0.9826
Badpix1	0.7727	0.7273	0.8673	0.8869	0.6453

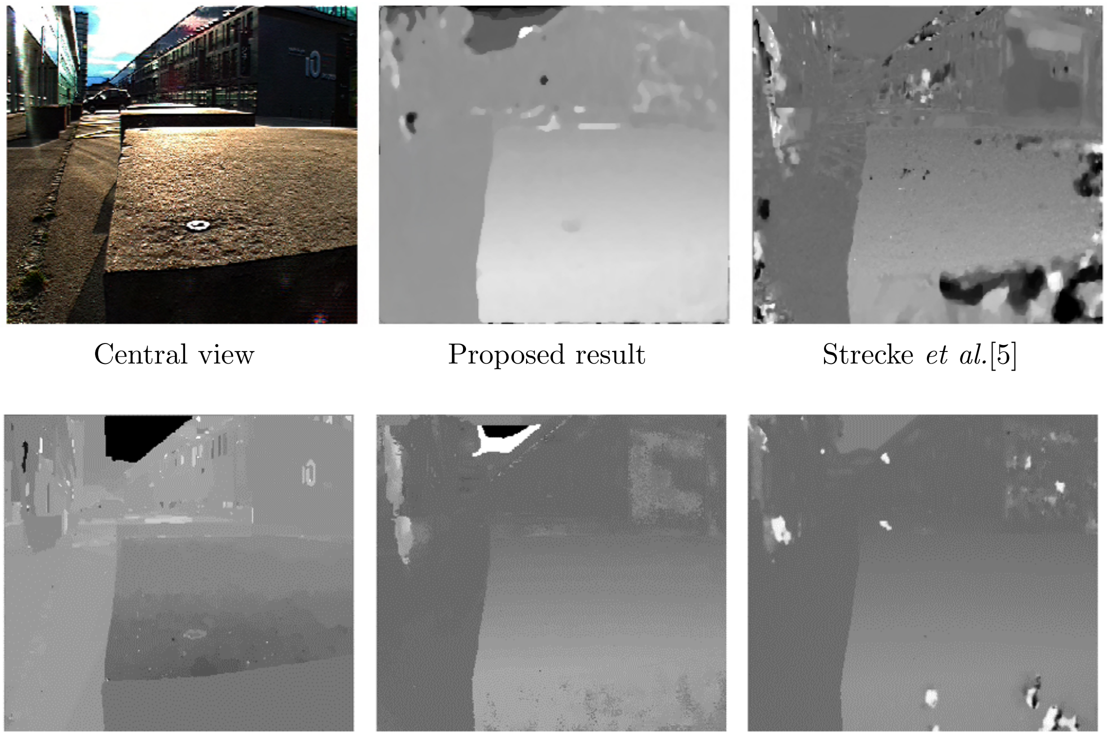
image, Fig. 5.22 (E), is a complex image with leaves and branches, but the proposed algorithm performs similarly to Strecke *et al.* [5] and Zhang *et al.* [7]. On the other hand, with Wang *et al.* [6] major parts of the image are misdetected. For the 'perforated metal 1' image Fig. 5.22 (A), parts of the image in the far background and foreground are represented with less error as compared to all the other depth maps, whereas in the 'perforated metal 3' image Fig. 5.22 (B), Wang *et al.* [6] better estimates the depth around the holes in the metal frame. In Fig. 5.21 and Fig. 5.22, misdetections can been seen for textureless regions for the 'backlight 1' image Fig. 5.21 (B), the 'perforated metal' image Fig. 5.22 (A), and the 'university' image Fig. 5.22 (D) i.e. the regions of the image with the sky.

5.4.3 Noisy image analysis

The results in Fig. 5.15, Fig. 5.23 and Table 5.6 demonstrate that the proposed algorithm is more noise-resilient than existing approaches. To further explore our algorithm's noise

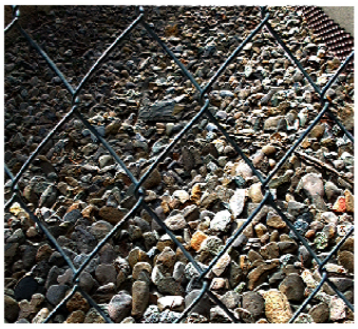


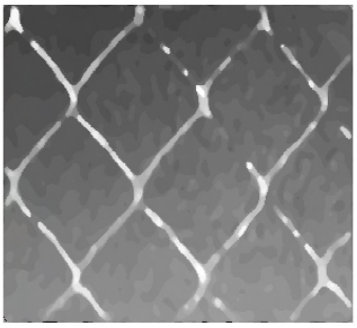
(A) ‘Danger de mort’ image

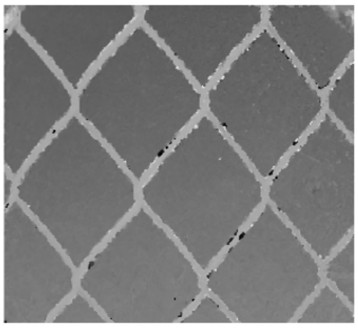


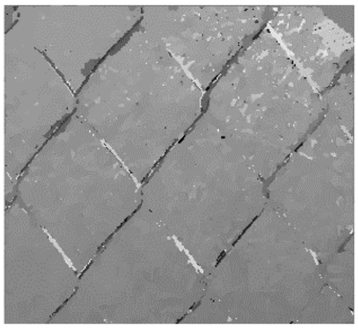
(B) ‘Backlight 1’ image

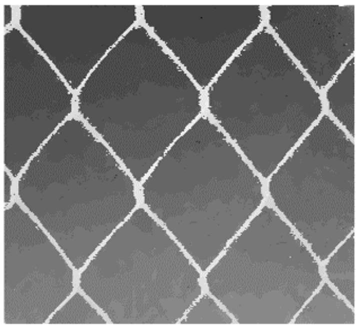
FIGURE 5.21: *Cont.*

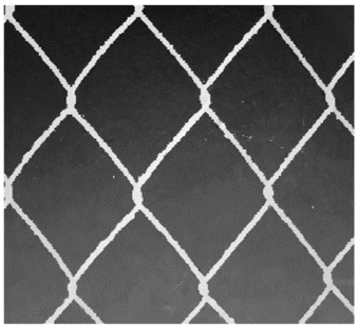

Central view


Proposed result

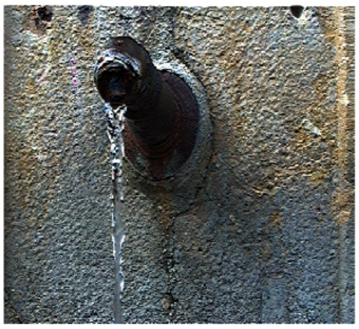

Strecke *et al.*[5]

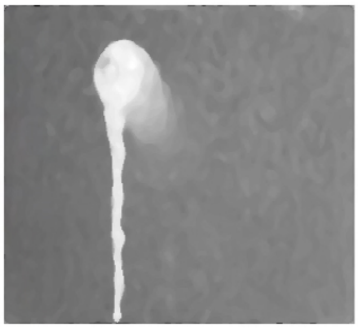

Wang *et al.* [6]

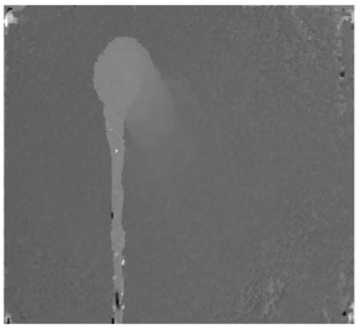

Zhang *et al.* [7]

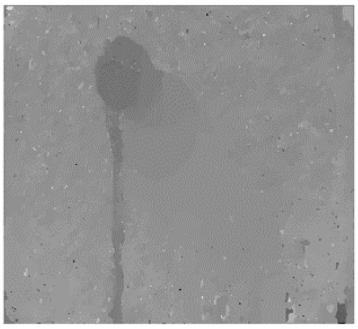

Shin *et al.* [8]

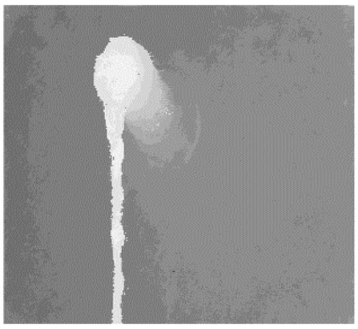
(C) ‘Chain link fence 2’ image

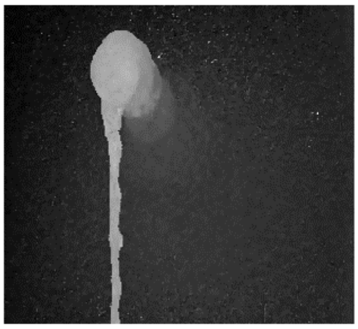

Central view


Proposed result


Strecke *et al.*[5]

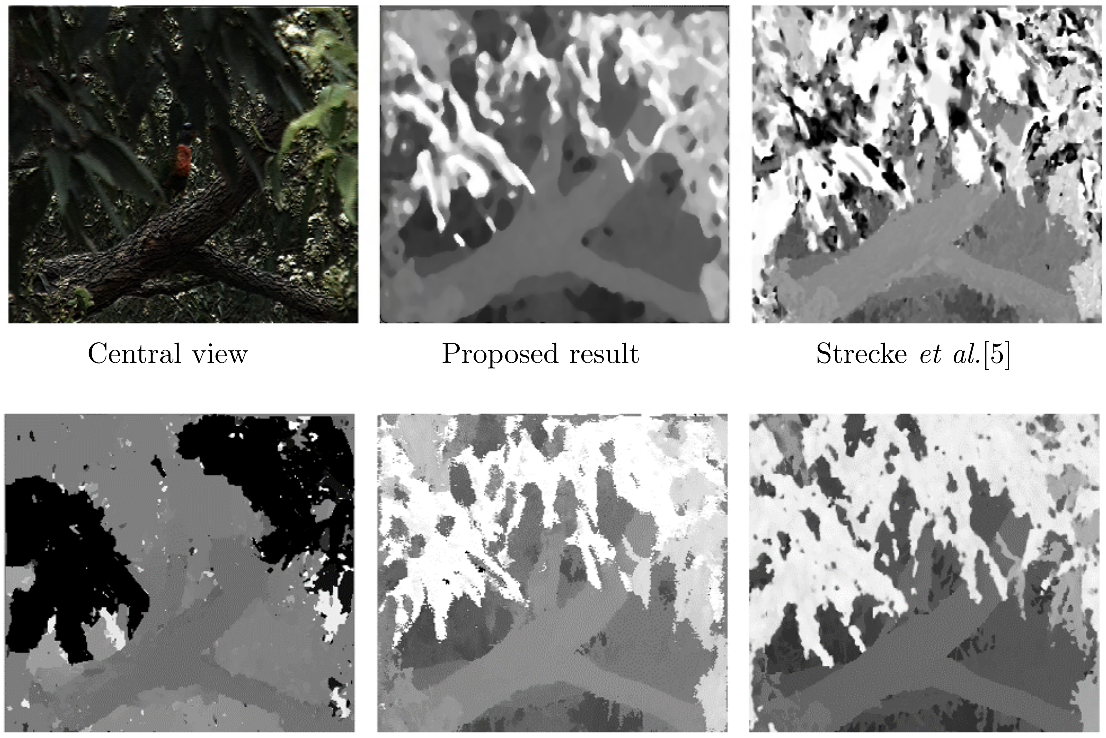

Wang *et al.* [6]


Zhang *et al.* [7]


Shin *et al.* [8]

(D) ‘Fountain 2’ image

FIGURE 5.21: *Cont.*



(E) 'Lorikeet' image

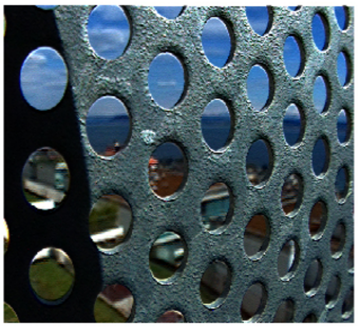
FIGURE 5.21: Visual comparison of the proposed algorithm with Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for real LF images.

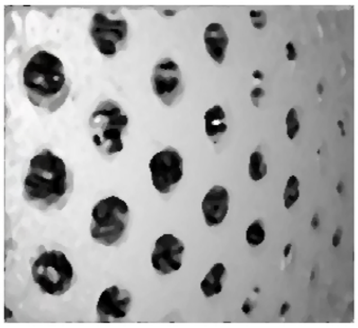
TABLE 5.6: Comparison for the 'Dot' image with ground truth for the proposed and state-of-the-art algorithms

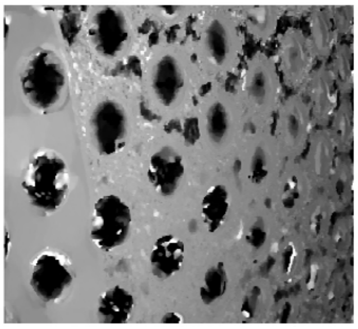
	Dots				
	Proposed result	Strecke *et al.* [5]	Wang *et al.*[6]	Zhang *et al.* [7]	Shin *et al.* [8]
Badpix7	**0.9605**	0.6273	0.8800	0.7357	0.9473
Badpix3	**0.8853**	0.4514	0.2485	0.4810	0.7957
Badpix1	0.3880	0.1777	0.0456	0.4810	0.5122

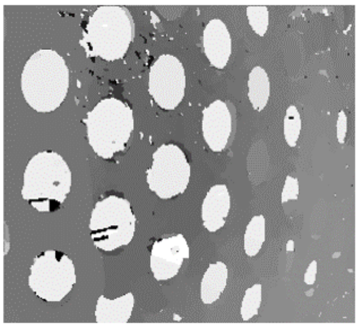
resilience, Gaussian noise approximates thermal and shot noise in images, and Gaussian noise with zero mean and variance of 0.01 was added to the 4D Light Field Dataset [3]. Gaussian noise was chosen as it approximates thermal and shot noise in images, as at large light levels, the Poisson distribution that describes shot noise approaches a normal distribution and can be approximated using Gaussian noise.

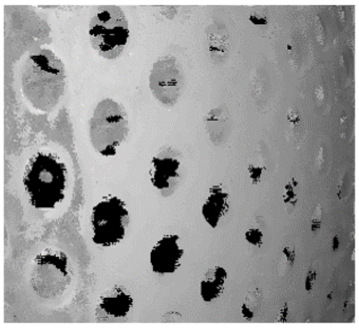
The proposed algorithm out-performs the benchmark algorithms for all images in all three

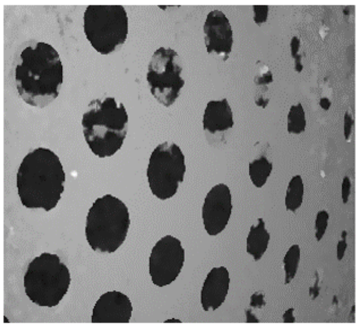

Central view


Proposed result

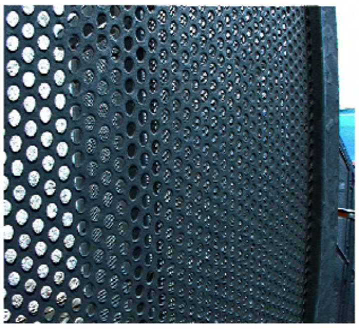

Strecke *et al.*[5]

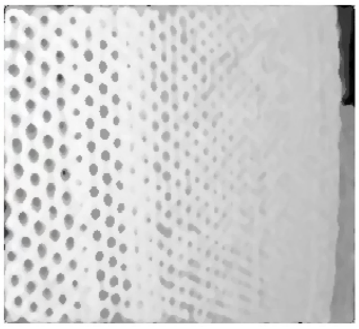

Wang *et al.* [6]

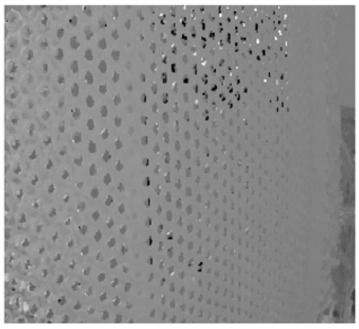

Zhang *et al.* [7]

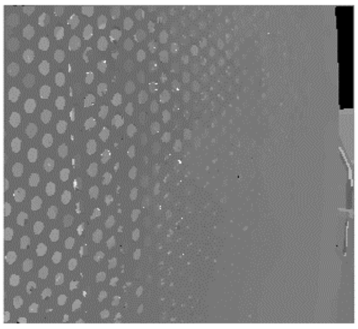

Shin *et al.* [8]

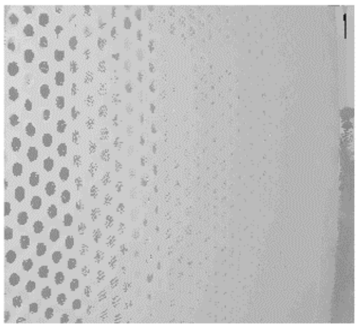
(A) 'Perforated metal 1' image

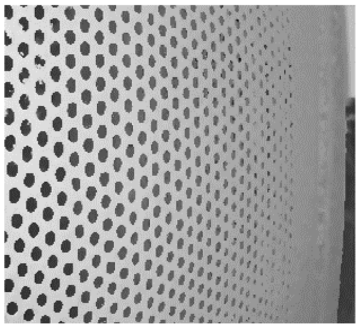

Central view


Proposed result


Strecke *et al.*[5]

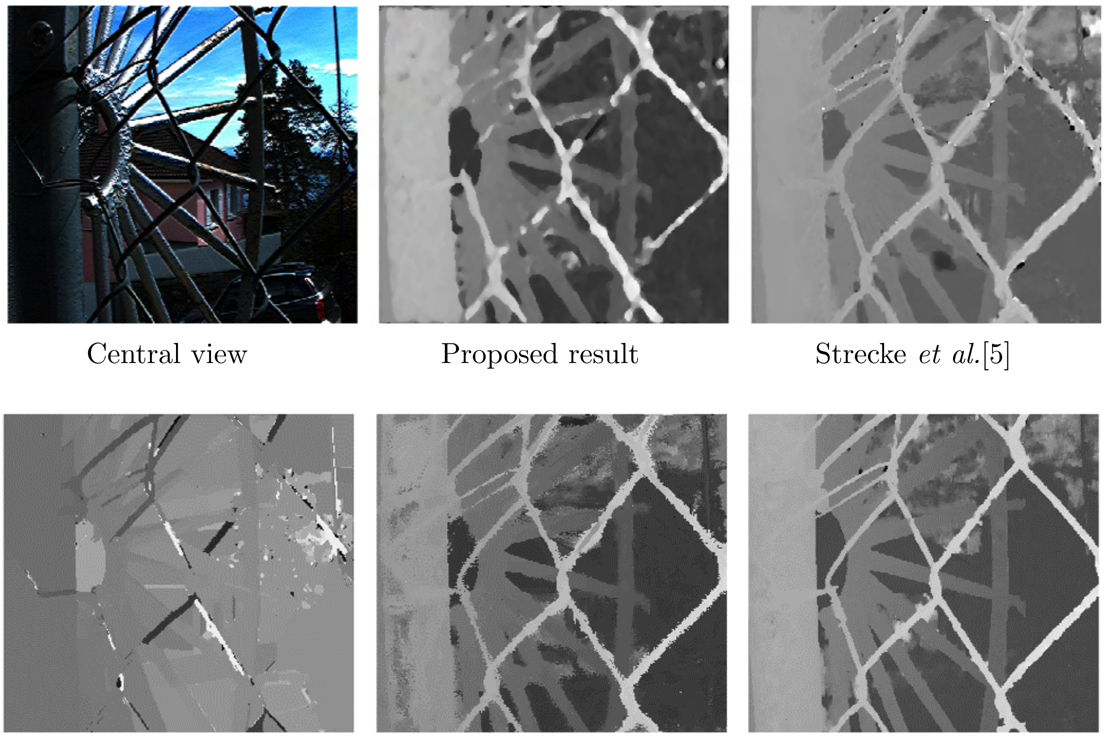

Wang *et al.* [6]


Zhang *et al.* [7]

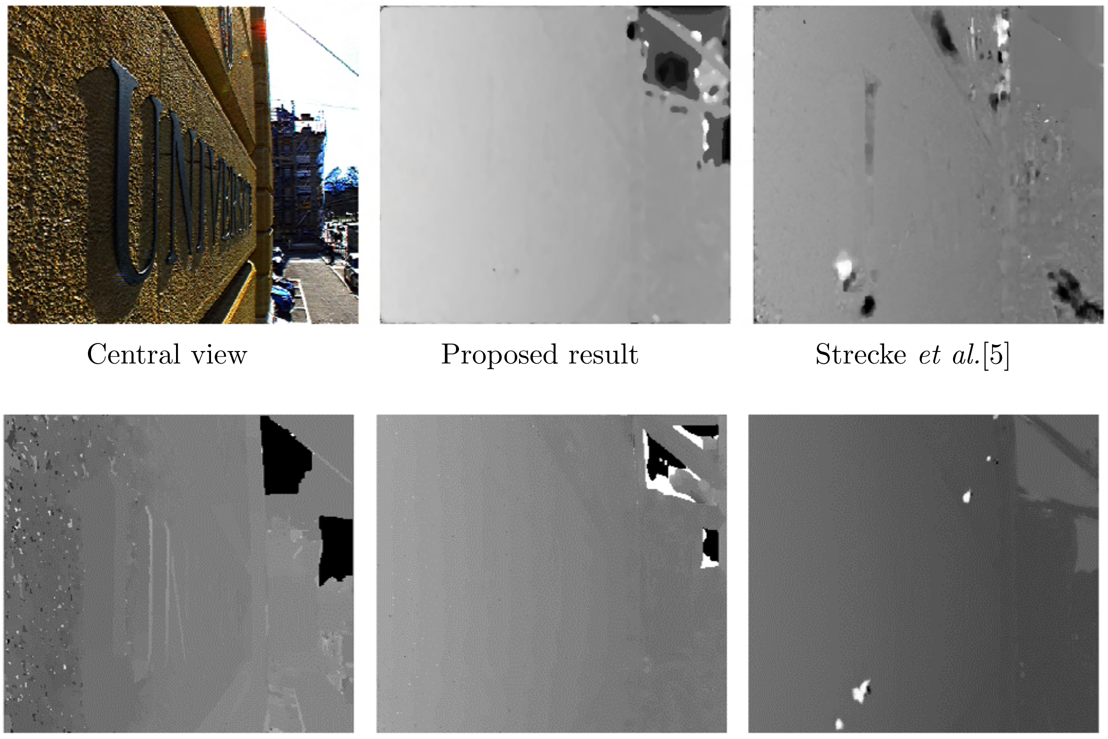

Shin *et al.* [8]

(B) 'Perforated metal 3' image

FIGURE 5.22: *Cont.*

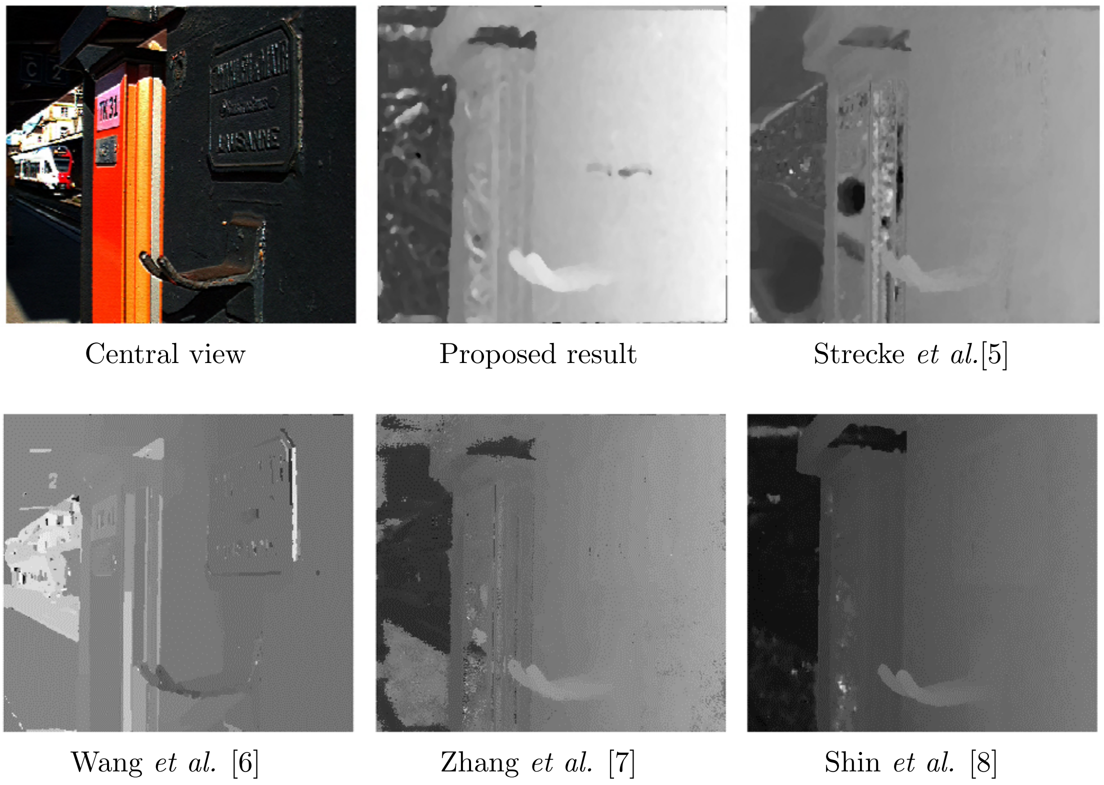


(C) ‘Spear fence 1’ image



(D) ‘University’ image

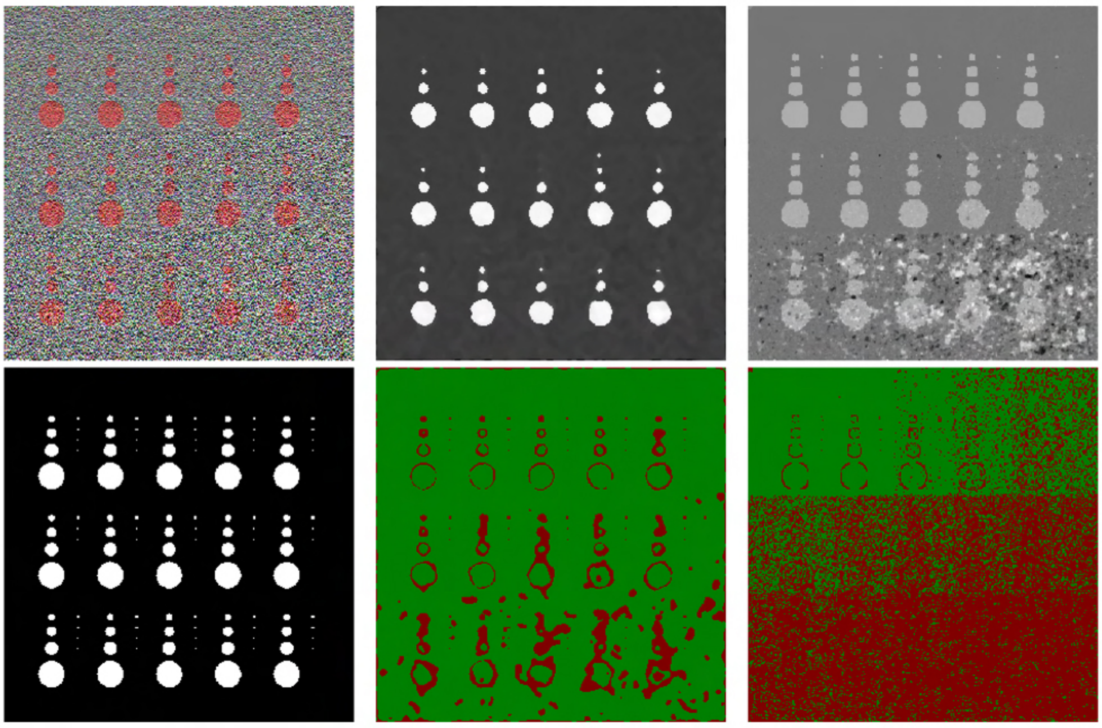
FIGURE 5.22: *Cont.*



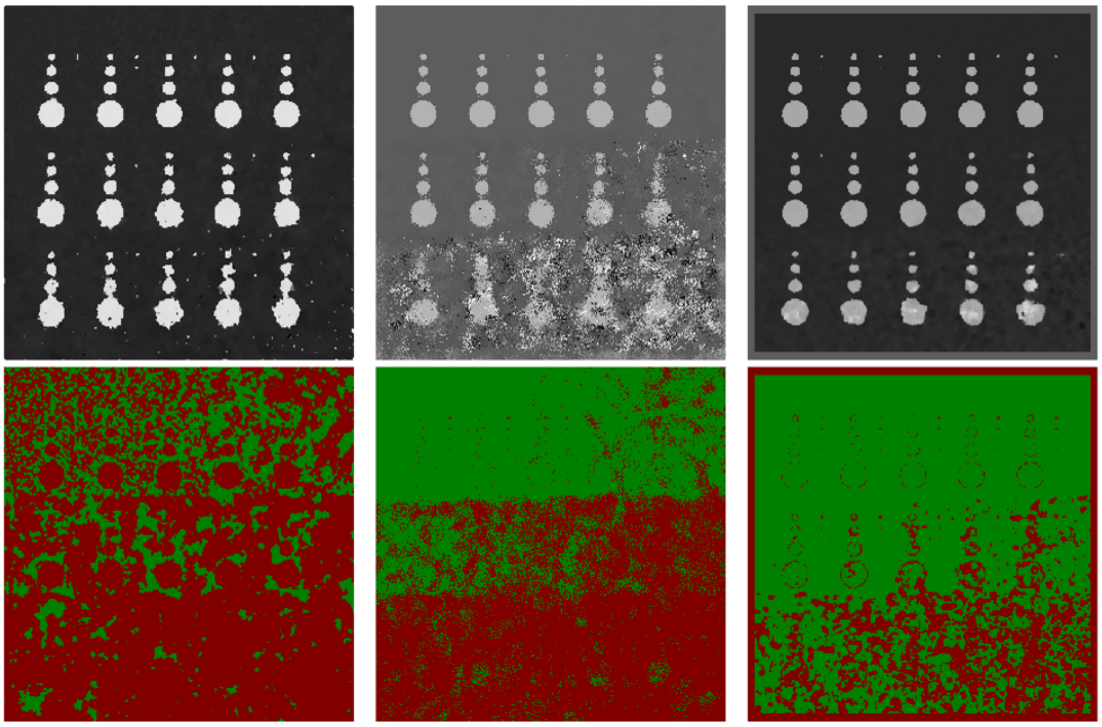
(E) 'Zwahlen and Mayr' image

FIGURE 5.22: Visual comparison of the proposed algorithm with Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for real LF images.

criteria as shown in Table 5.7, Table 5.8, Fig. 5.24 and Fig. 5.25. The algorithm from Zhang *et al.* [7] and Wang *et al.* [6] generate disparity maps that use the maximum and minimum depth values from the ground-truth depth map to scale the disparity map accordingly. The problem with noisy images is that misdetections and outliers make the re-scaling unreliable and nontrivial. It is clear from the results shown in Table 5.7 and Table 5.8 that the accuracy of the proposed algorithm is also significantly affected for most of the images, but the algorithm is still able to estimate a depth map with comparatively high accuracy compared to the state-of-the-art algorithms. The average accuracy for our algorithm for Badpix 0.07 across the ten images used for testing shown in Fig. 5.24 and Fig. 5.25 is 0.75, whereas for Shin *et al.* [8], Strecke *et al.* [5], Wang *et al.* [6] and Zhang *et al.* [7] it is 0.6, 0.3, 0.26 and 0.11 respectively. For the 'dots', and 'pyramids' images the accuracy is over 95% out-performing the state-of-the-art algorithms. The images with finer details like the 'backgammon' image in Fig. 5.24(A) has a considerable



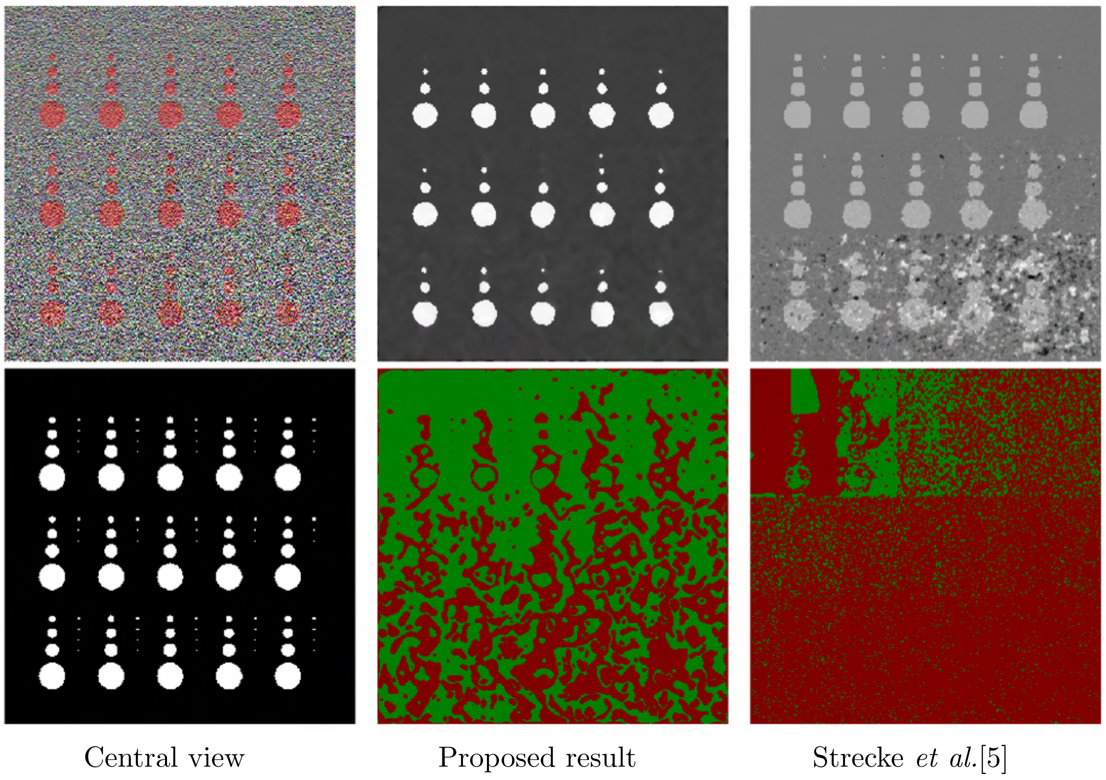
Central view Proposed result Strecke *et al.*[5]

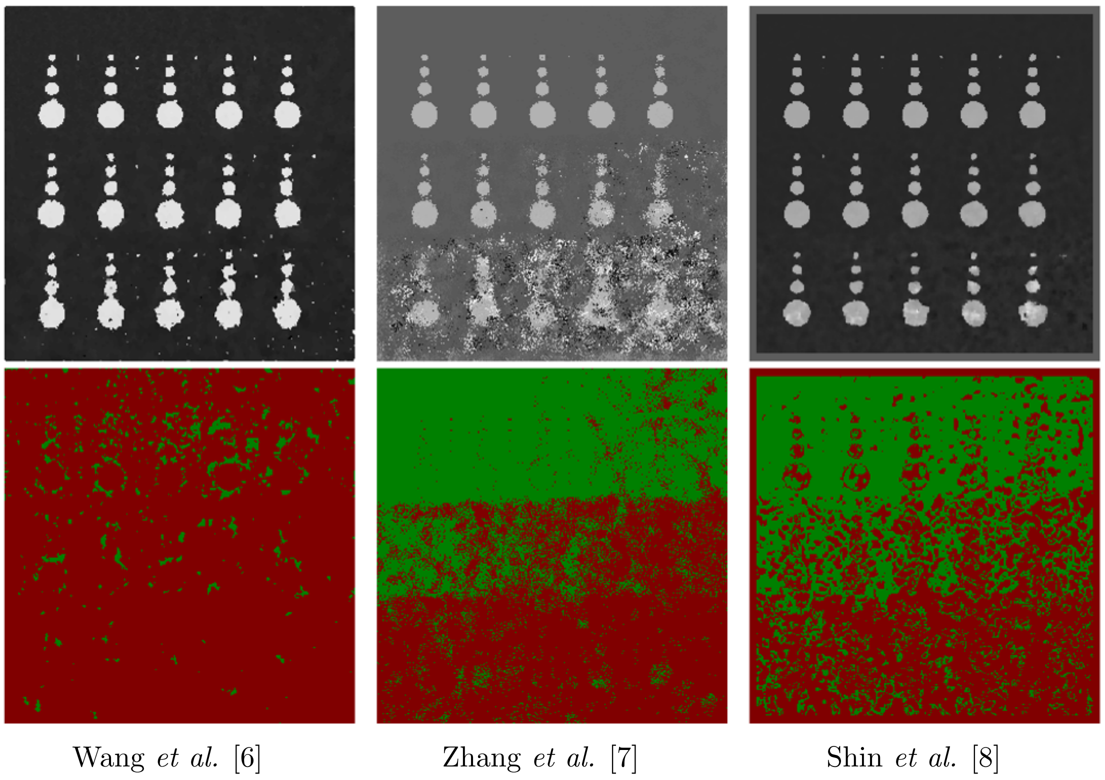


Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(A) ‘Dots’ image - Badpix 0.03

FIGURE 5.23: *Cont.*





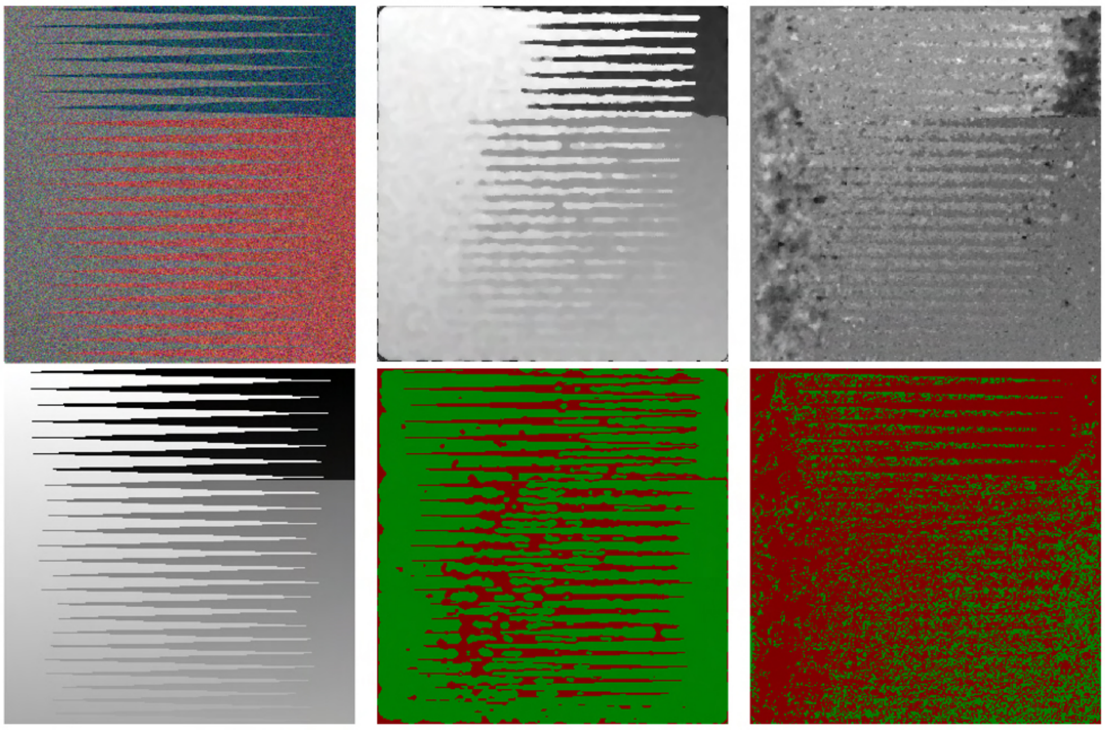
(B) ‘Dots’ image - Badpix 0.01

FIGURE 5.23: Visual comparison of the ground truth with the proposed algorithm, Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for dots image for Badpix 0.03 and 0.01.

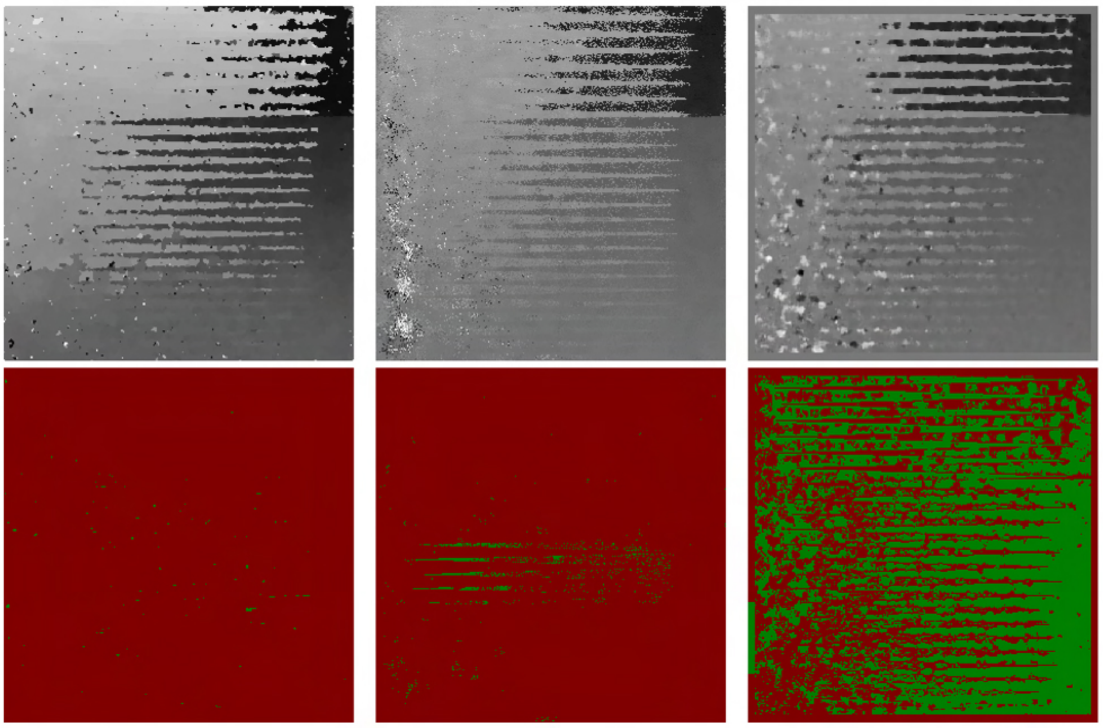
Table 5.7: Quantitative depth map comparison to ground truth for synthetic data for noisy images

	Back-gammon	Dots	Kitchen	Medi-eval2	Museum
			Proposed Results		
Badpix7	**0.7408**	**0.9620**	**0.6341**	**0.8171**	**0.6921**
Badpix3	**0.5126**	**0.8561**	**0.4323**	**0.5584**	**0.4889**
Badpix1	**0.2101**	0.2808	**0.1733**	**0.2290**	**0.2023**
			Strecke *et al.* [5]		
Badpix7	0.2781	0.3975	0.2309	0.3024	0.1938
Badpix3	0.1312	0.1895	0.1140	0.1419	0.0919
Badpix1	0.0450	0.0634	0.0402	0.0487	0.0318
			Wang *et al.* [6]		
Badpix7	0.0022	0.8619	0.1229	0.0892	0.1868
Badpix3	0.0008	0.7684	0.0570	0.0340	0.0790
Badpix1	0.0003	0.1351	0.0197	0.0083	0.0260
			Zhang *et al.* [7]		
Badpix7	0.0144	0.0002	0.1587	0.2456	0.1054
Badpix3	0.0057	0.0001	0.0666	0.1022	0.0517
Badpix1	0.0019	0.0000	0.0217	0.0322	0.0174
			Shin *et al.* [8]		
Badpix7	0.5778	0.8990	0.5035	0.6512	0.5237
Badpix3	0.3265	0.6624	0.3090	0.3898	0.3112
Badpix1	0.1162	0.3034	0.1247	0.1451	0.1181

amount of misdetections, but algorithm is still able to obtain a Badpix 0.07 value of 0.74. The 'kitchen' and 'museum' image in Fig. 5.24(C) and (E) shows errors for transparent or reflective surfaces, but the depth map for our algorithm estimates the background and foreground region in the image accurately with Badpix 0.07 values of 0.63 and 0.69 respectively, whereas the depth map from Wang *et al.* [6] and Zhang *et al.* [7] produces errors in the background and foreground regions and a Badpix 0.07 value below 0.25. Shin *et al.* [8], on the other hand, shows a Badpix 0.07 value 0.5 and 0.52, but large parts of the background and foreground region are misdetected. An important observation that can be drawn from Table. 5.4, Table. 5.5, Table. 5.7 and Table. 5.8 is that without the added noise, the average accuracy for the proposed algorithm, Shin *et al.* [8], Strecke *et al.* [5], Wang *et al.* [6] and Zhang *et al.* [7] is 0.96, 0.9, 0.89, 0.79 and 0.86, respectively. After the noise is added to the images, our accuracy reduces to 0.75, whereas for Shin *et al.* [8] it reduces to 0.6, Strecke *et al.* [5] it reduces to 0.3, Wang *et al.* [6] it reduces to 0.26 and Zhang *et al.* [7] it reduces to 0.11.



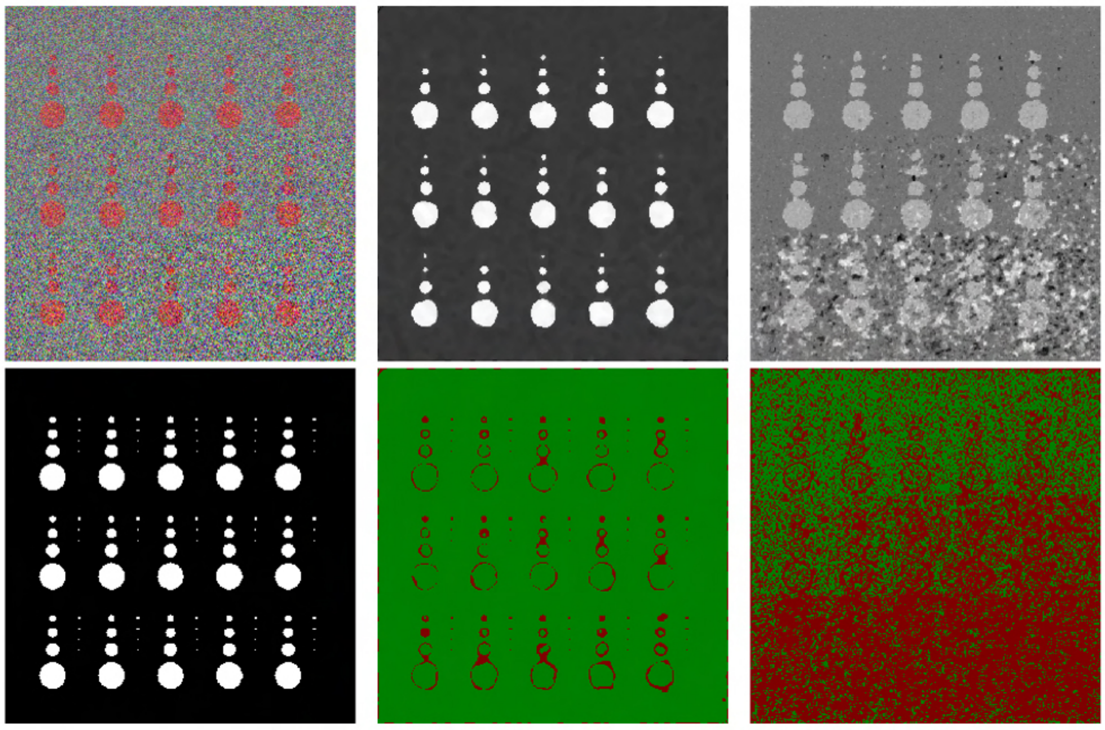
Central view Proposed result Strecke *et al.*[5]



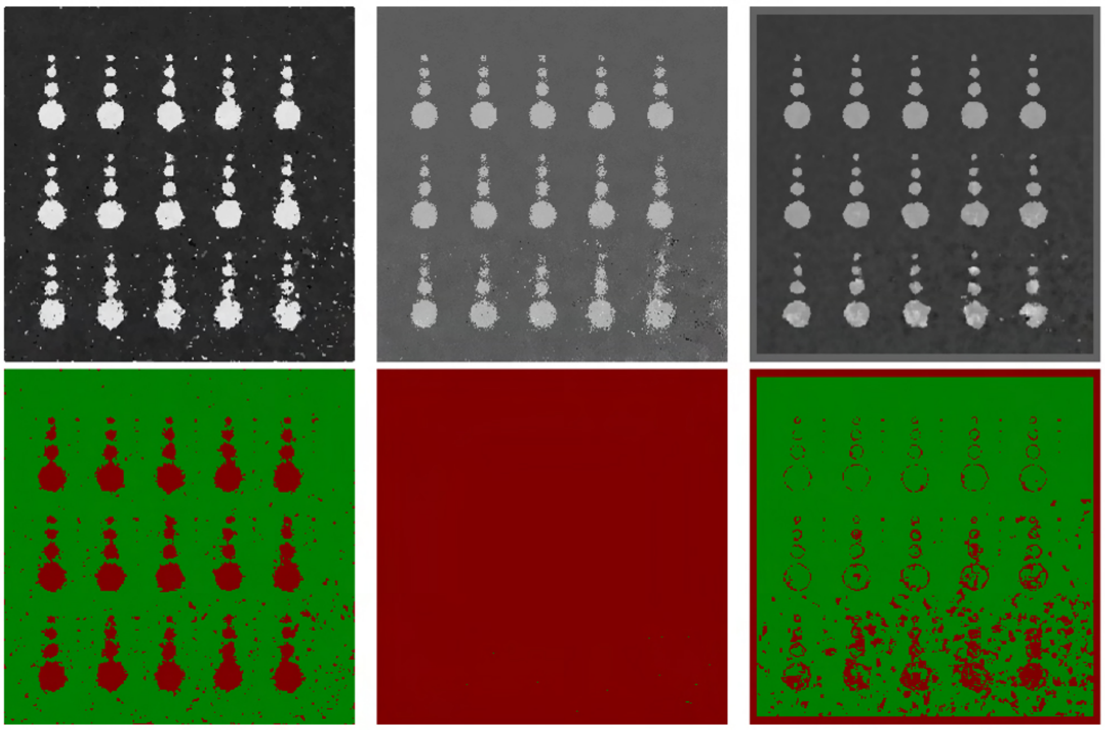
Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(A) ‘Backgammon’ image

FIGURE 5.24: *Cont.*



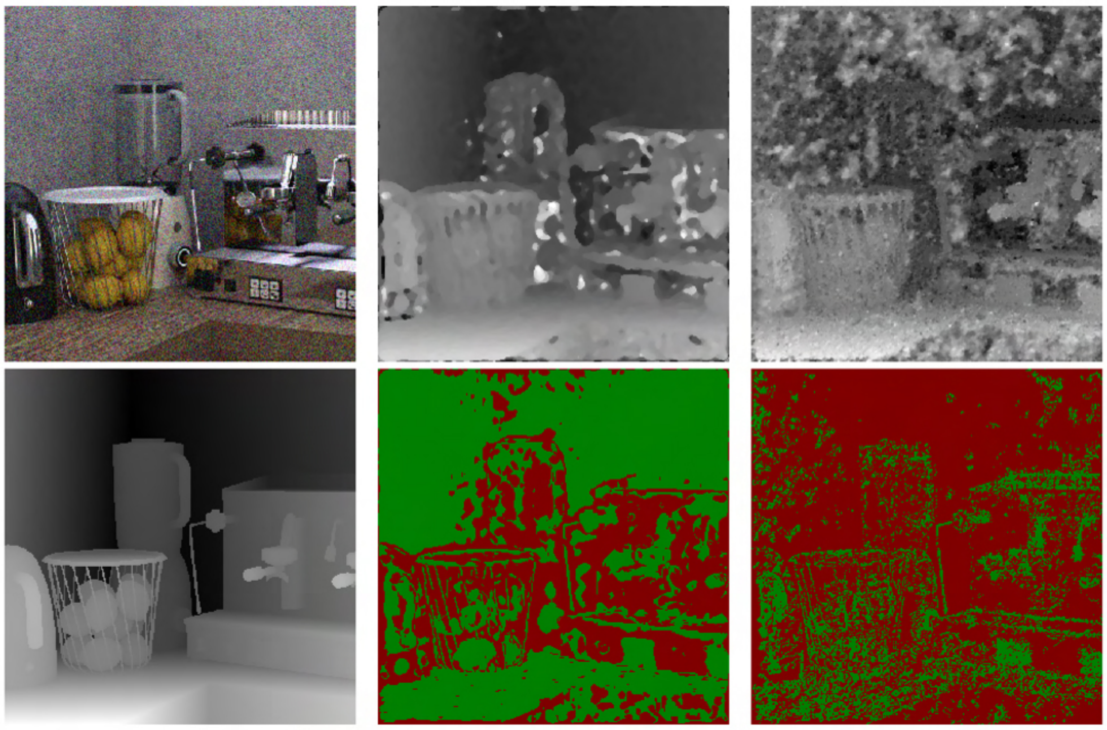
Central view Proposed result Strecke *et al.*[5]



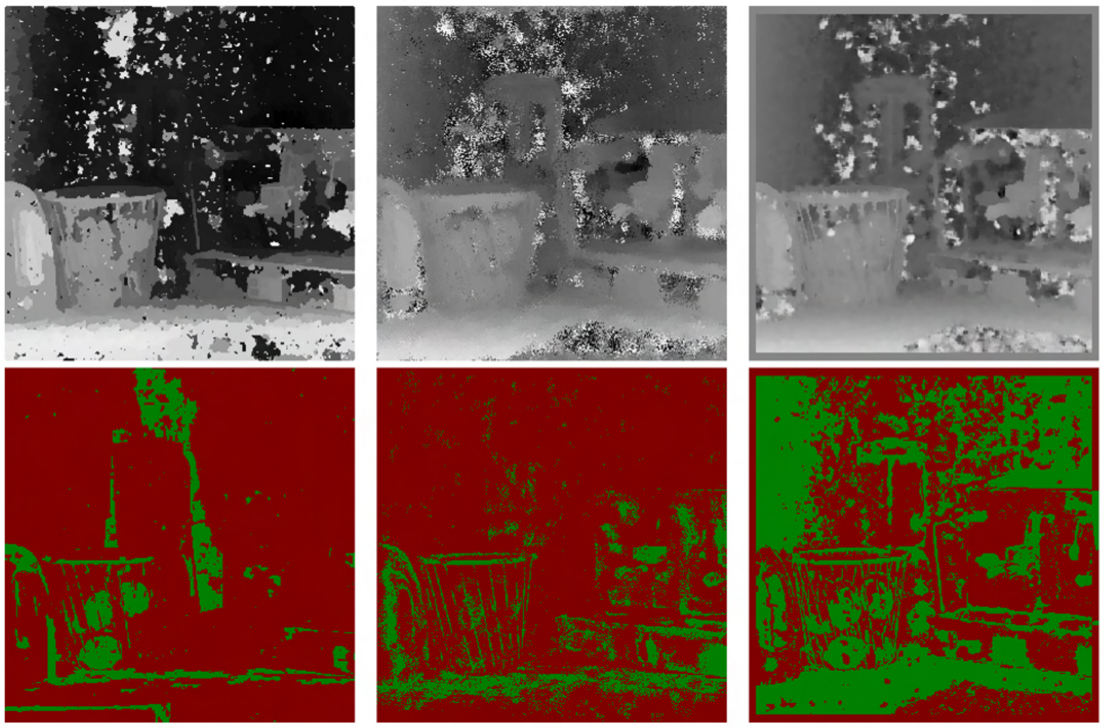
Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(B) ‘Dots’ image

Figure 5.24: *Cont.*



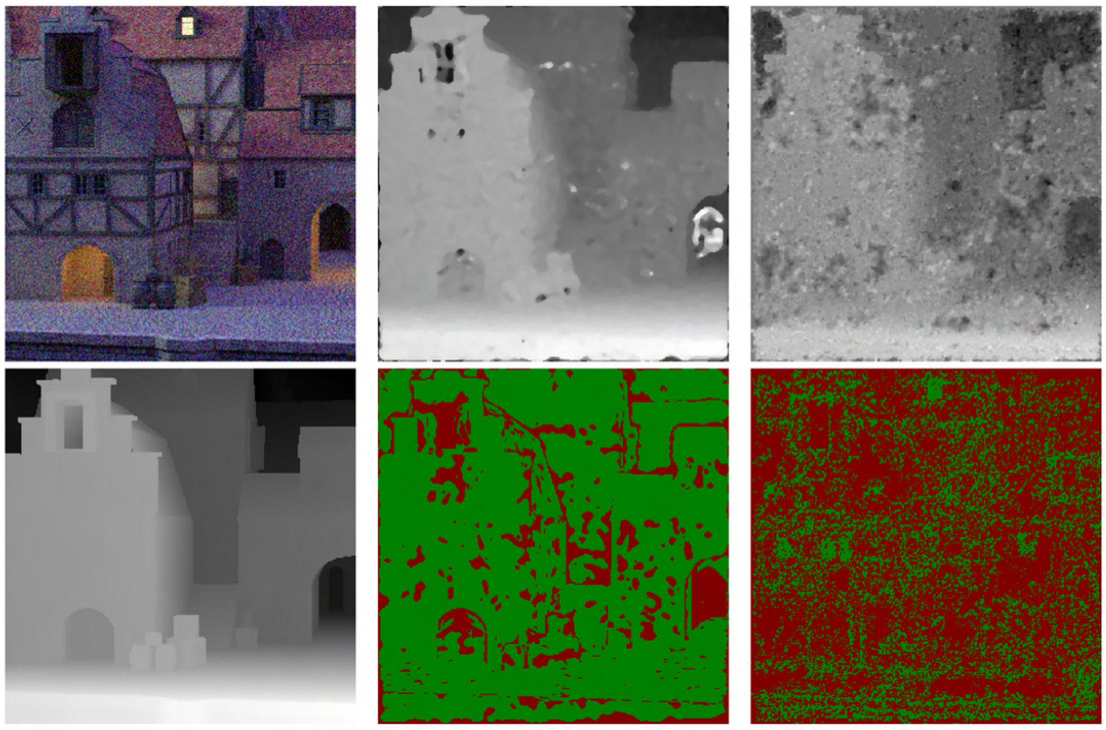
Central view Proposed result Strecke *et al.*[5]



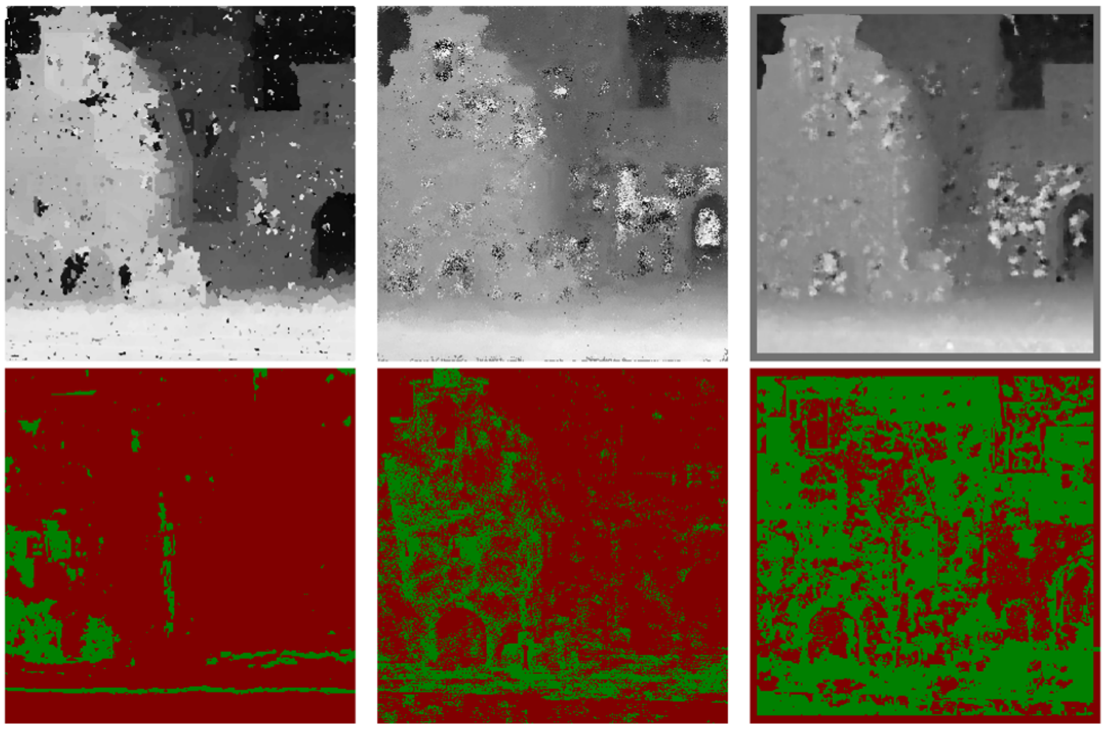
Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

(C) 'Kitchen' image

FIGURE 5.24: *Cont.*



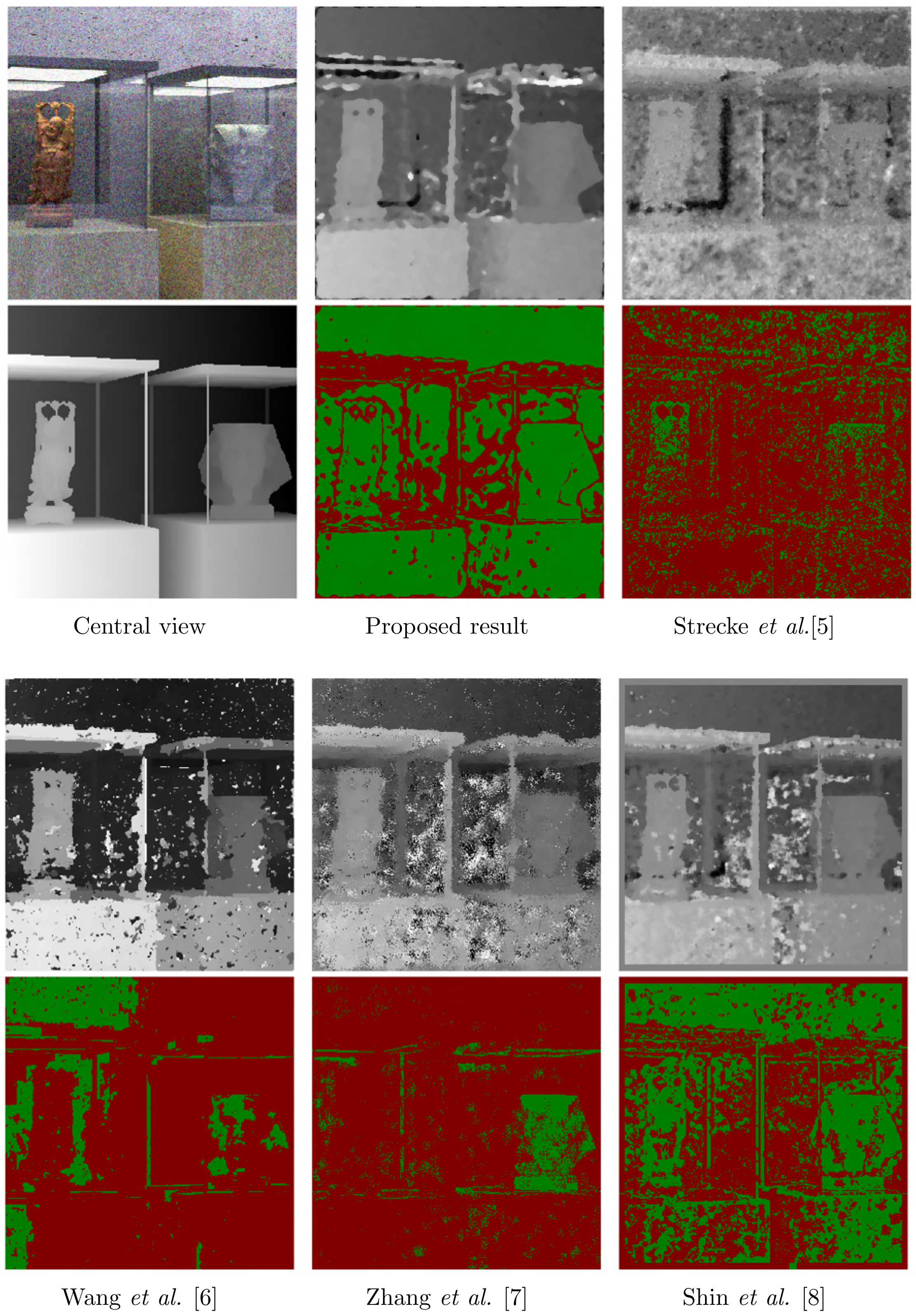
Central view Proposed result Strecke *et al.*[5]



Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

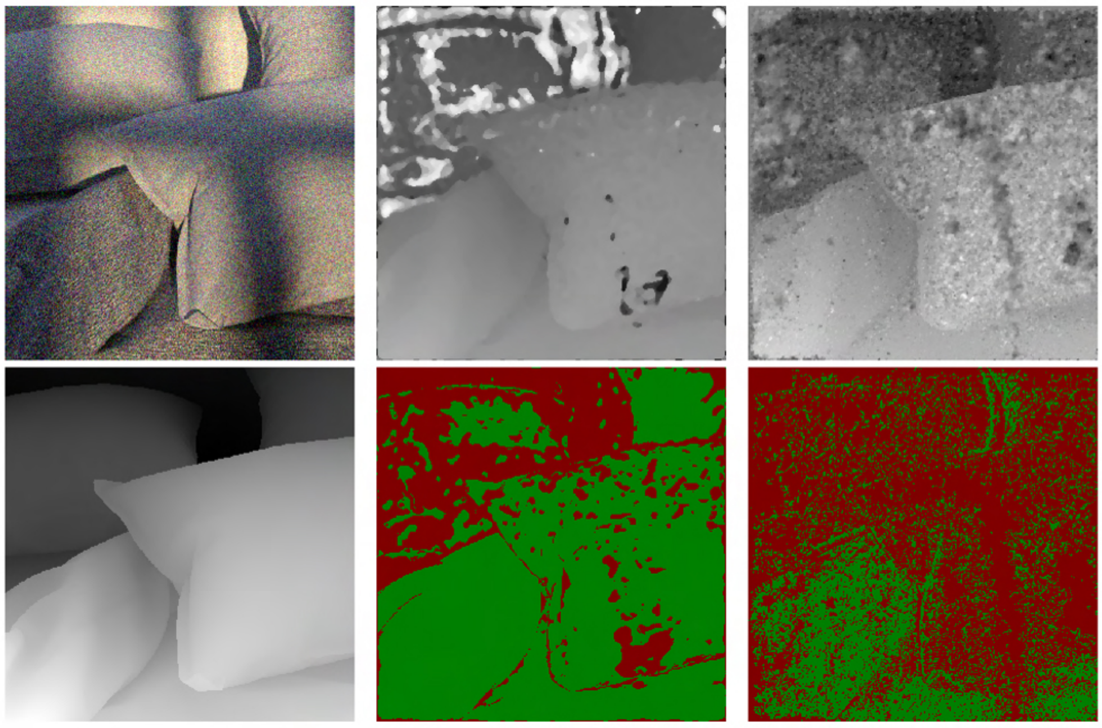
(D) 'Medieval 2' image

FIGURE 5.24: *Cont.*

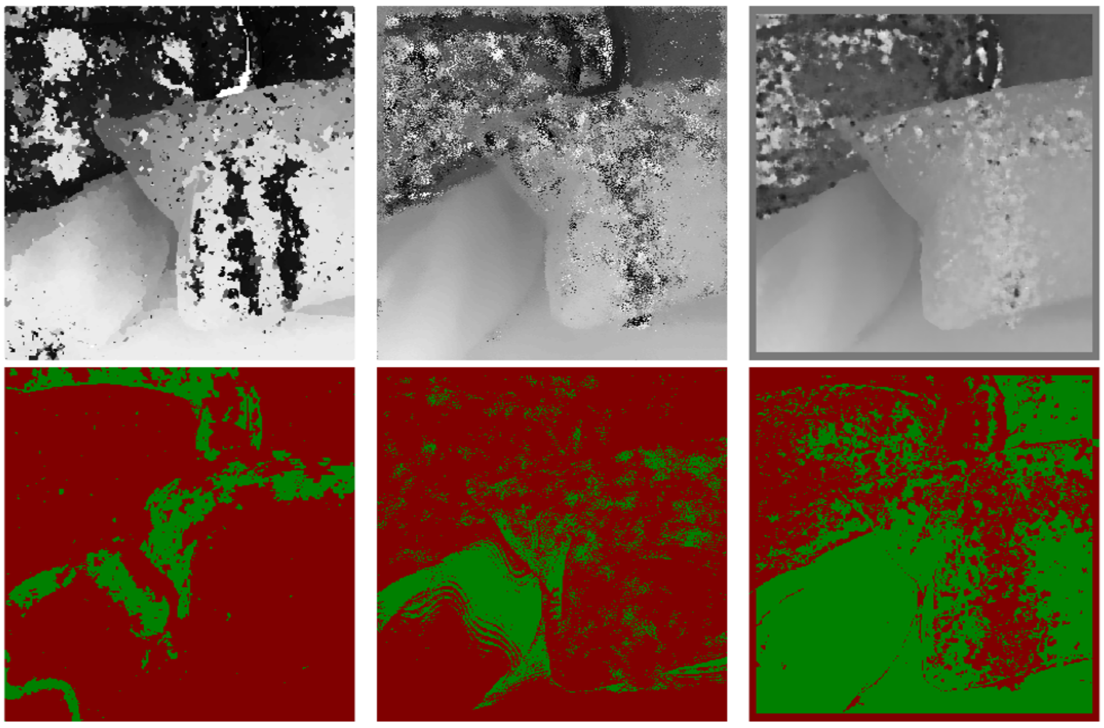


(E) 'Museum' image

FIGURE 5.24: Visual comparison of the proposed algorithm with Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for synthetic LF images with added noise.



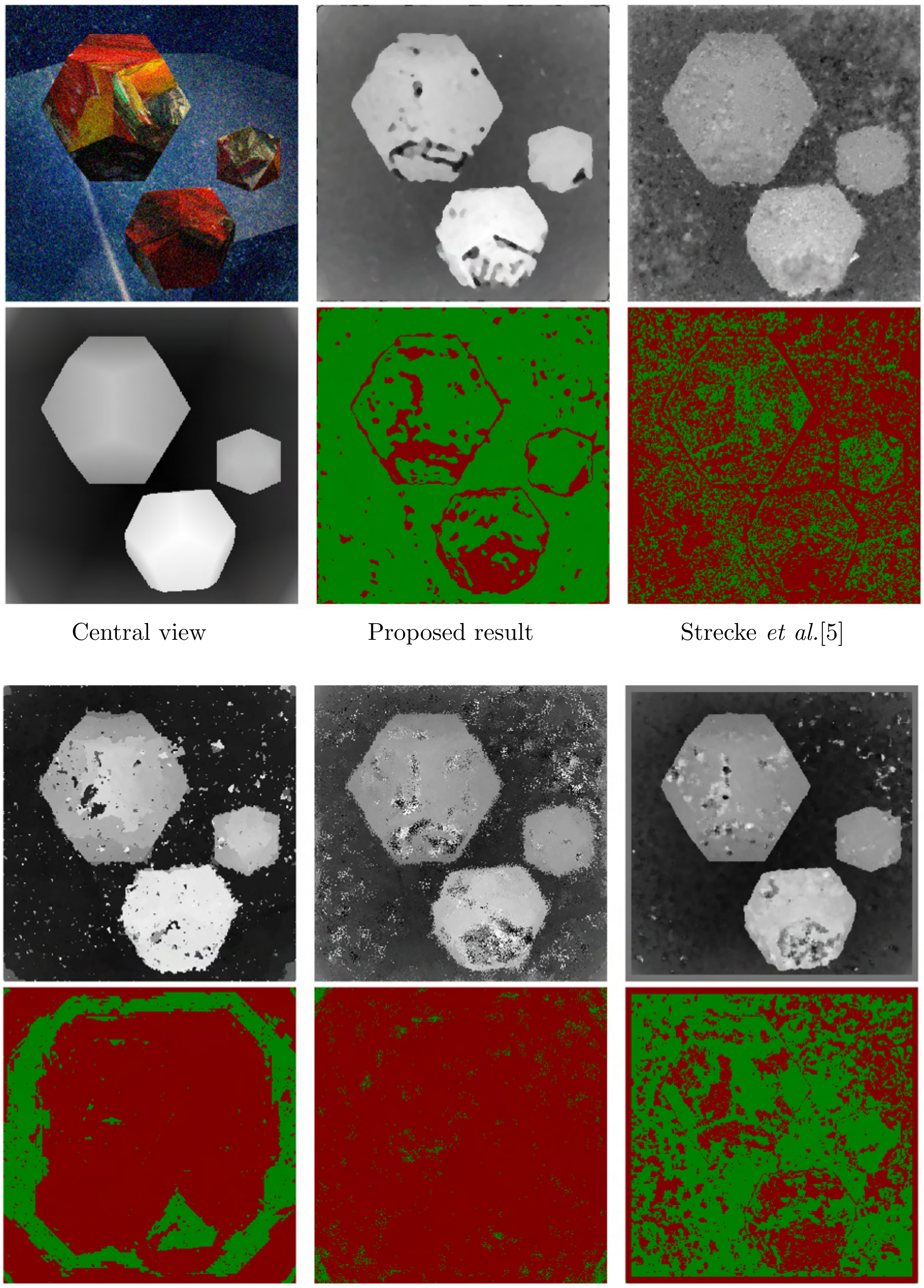
Central view Proposed result Strecke *et al.*[5]



Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

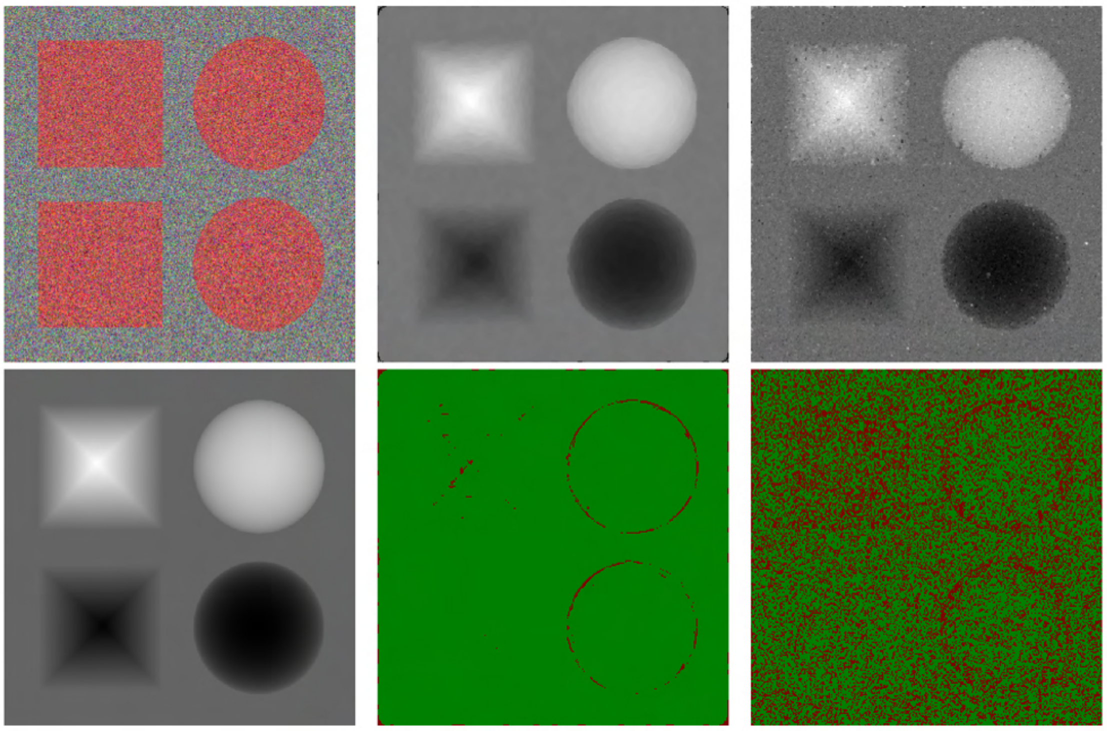
(A) ‘Pillows’ image

FIGURE 5.25: *Cont.*

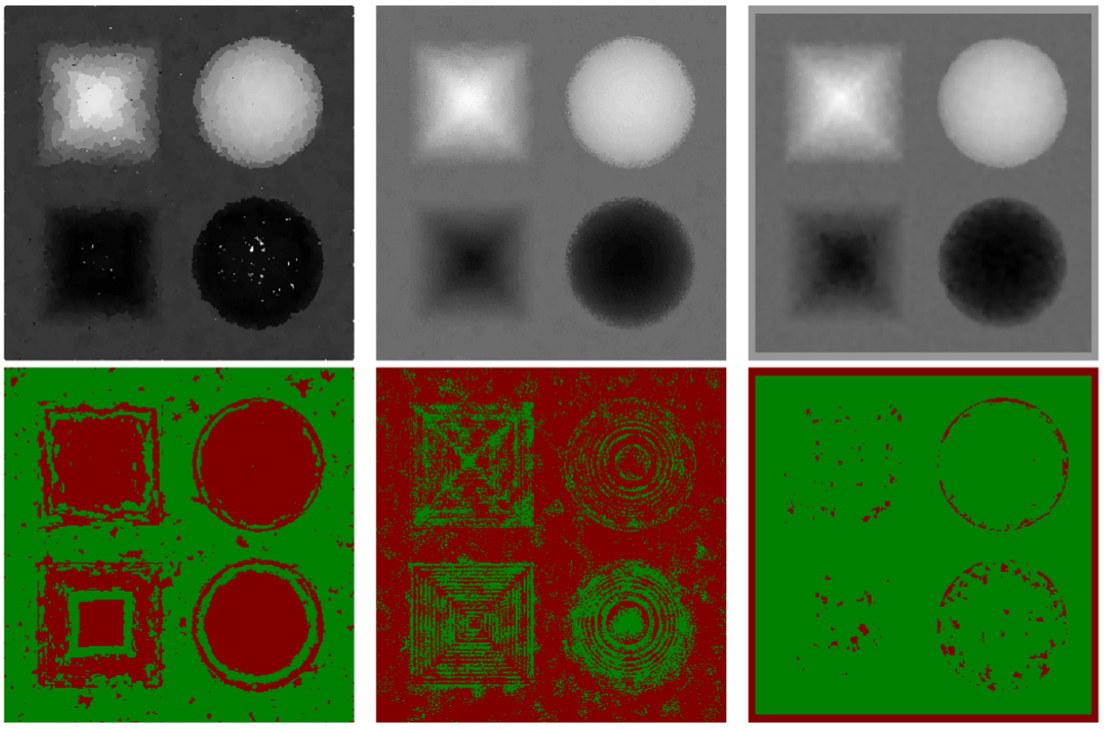


(B) 'Platonic' image

FIGURE 5.25: *Cont.*



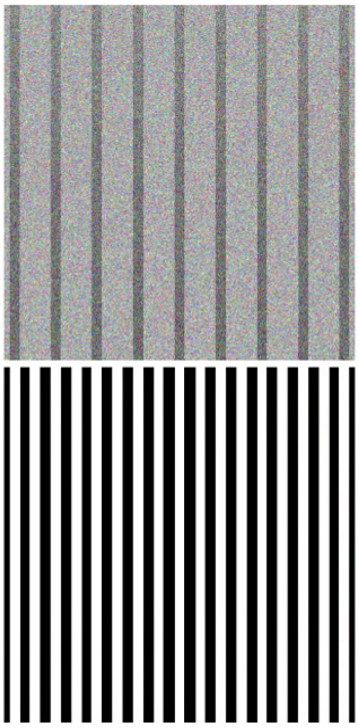
Central view Proposed result Strecke *et al.*[5]

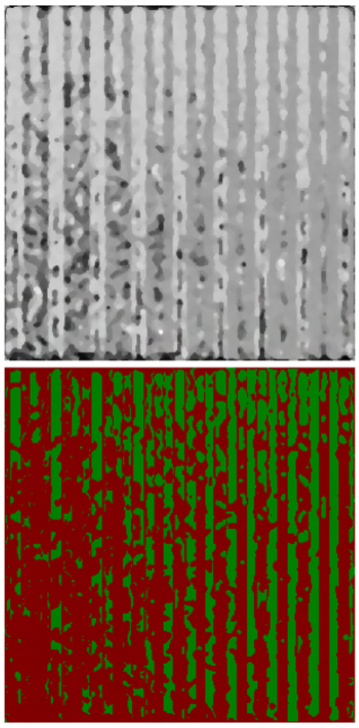


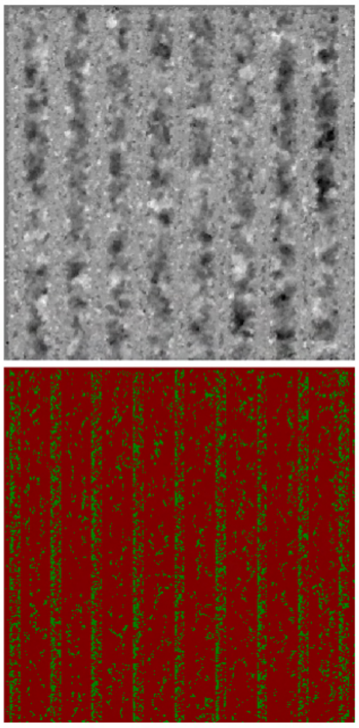
Wang *et al.* [6] Zhang *et al.* [7] Shin *et al.* [8]

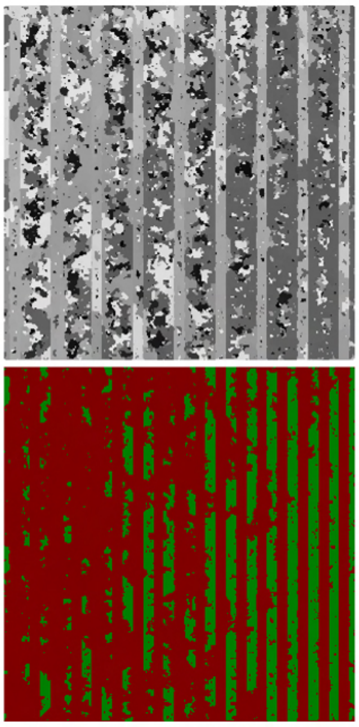
(C) 'Pyramids' image

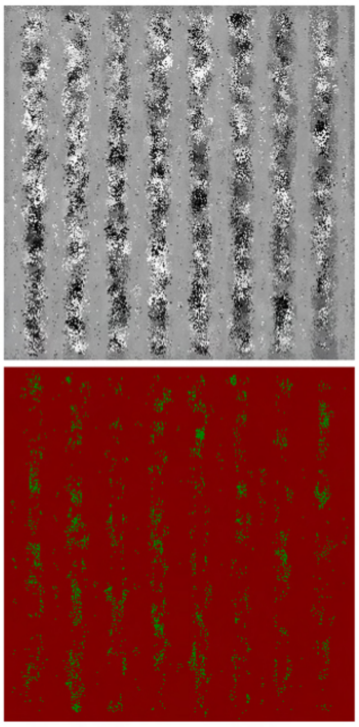
Figure 5.25: *Cont.*

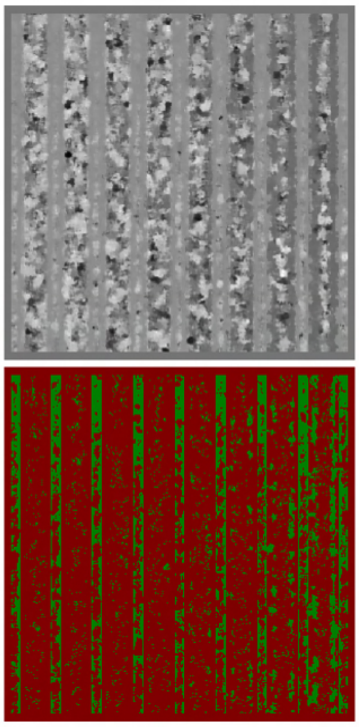

Central view


Proposed result


Strecke *et al.*[5]

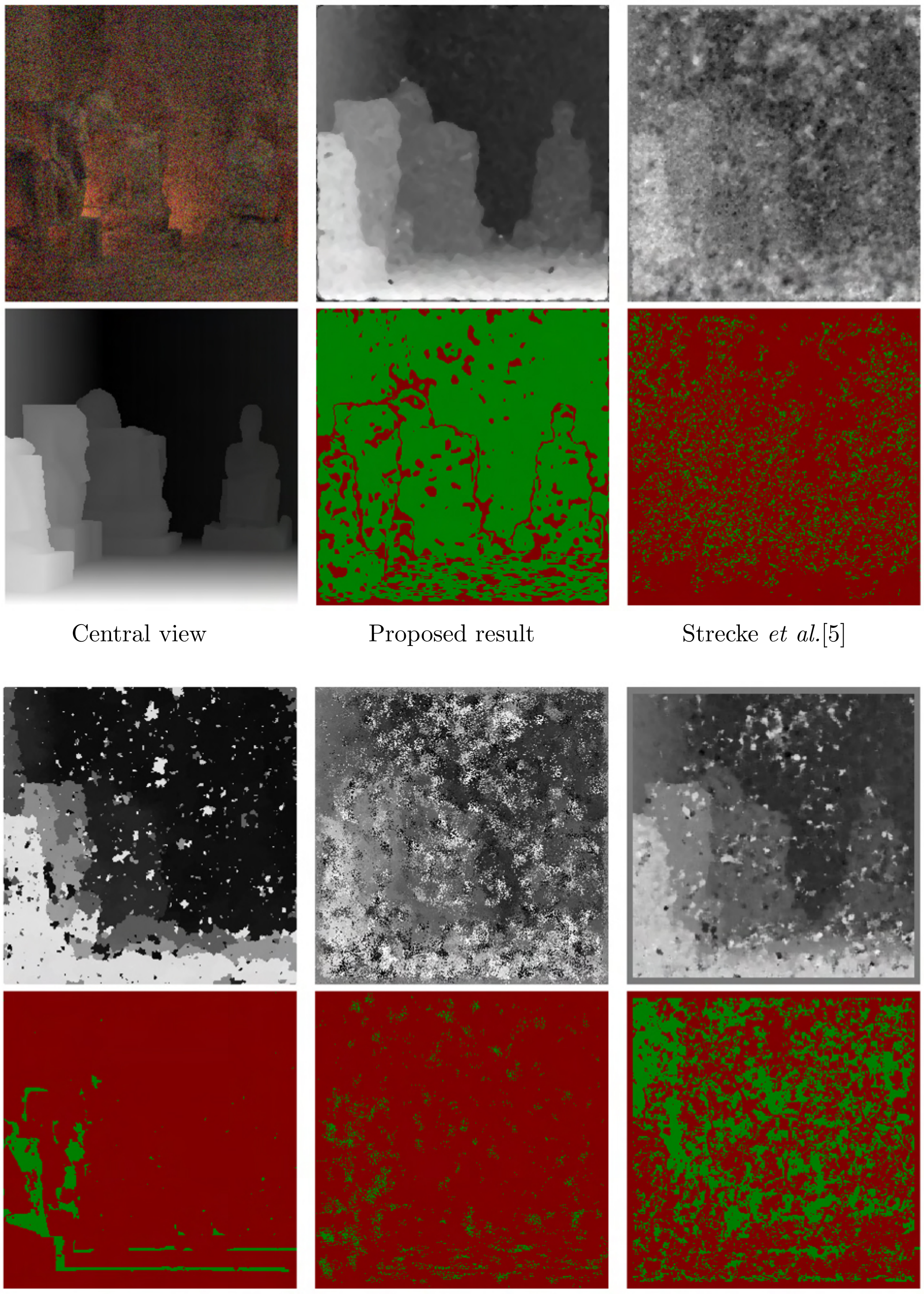

Wang *et al.* [6]


Zhang *et al.* [7]


Shin *et al.* [8]

(D) 'Stripes' image

FIGURE 5.25: *Cont.*



(E) 'Tomb' image

FIGURE 5.25: Visual comparison of the proposed algorithm with Strecke *et al.* [5], Wang *et al.* [6], Zhang *et al.* [7] and Shin *et al.* [8] for synthetic LF images with added noise.

Table 5.8: Quantitative depth map comparison to ground truth for synthetic data for noisy images

	Pillows	Platonic	Pyramids	Stripes	Tomb
			Proposed Results		
Badpix7	**0.6823**	**0.8457**	**0.9891**	**0.3582**	**0.8008**
Badpix3	**0.5099**	**0.6272**	**0.9108**	**0.1982**	**0.5089**
Badpix1	**0.2417**	**0.2600**	**0.4835**	**0.0930**	**0.1877**
			Strecke *et al.* [5]		
Badpix7	0.3212	0.3093	0.6557	0.1388	0.1404
Badpix3	0.1698	0.1476	0.3456	0.0641	0.0615
Badpix1	0.0613	0.0511	0.1136	0.0212	0.0206
			Wang *et al.* [6]		
Badpix7	0.1472	0.2515	0.6136	0.2488	0.0561
Badpix3	0.0700	0.1126	0.0870	0.1106	0.0209
Badpix1	0.0257	0.0366	0.0141	0.0014	0.0069
			Zhang *et al.* [7]		
Badpix7	0.1794	0.0394	0.2569	0.0419	0.0530
Badpix3	0.0814	0.0148	0.0917	0.0219	0.0214
Badpix1	0.0273	0.0045	0.0115	0.0078	0.0075
			Shin *et al.* [8]		
Badpix7	0.6000	0.6621	0.9729	0.2084	0.3957
Badpix3	0.4383	0.3836	0.8325	0.1051	0.1904
Badpix1	0.2193	0.1461	0.4132	0.0372	0.0664

Table 5.9: Average runtime for synthetic data

Synthetic light field dataset				
Proposed	Strecke *et al.* [5]	Wang *et al.* [6]	Zhang *et al.* [7]	Shin *et al.* [8]
1418s	462s	243s	537s	7s

5.4.4 Runtime complexity analysis

The code for the proposed algorithm was implemented in MATLAB$^{\text{TM}}$ on an Intel i7 machine at 1.9GHz and 16 GB of RAM. The results for Strecke *et al.* [5], Wang *et al.* [6] and Zhang *et al.* [7] were generated using the same machine. Table 5.9 shows the average runtime over all the images in the dataset for a single run for the proposed algorithm compared to the state-of-the-art algorithms. As Shin *et al.* [8] propose a CNN approach, their network has to be trained and their network takes 5 days to train.

The runtime for the proposed algorithm varies for each of the images as the algorithm first calculates the maximum and minimum depths for the image and then generates the focal stack. The runtime for the algorithm can be divided into 4 stages: the first stage calculates the initial depth which takes an average of 52.18 sec; all of the pre-processing steps in the second stage for the images takes an average of 0.21 sec per image; the third stage generates the focal stack where the majority of the processing time is spent, an average duration of 2.17 sec per image; and the last stage is where the depth map is estimated and refined, which on average consumes a duration of 0.8 sec per image. For real images, the initial depth estimation stage takes an average of 150 sec; all the pre-processing steps for the images takes an average of 0.21 sec per image; the third stage generates the focal stack where majority of the processing time is spent, an average duration of 9.68 sec per image, and the last stage estimating and refining the depth map on average takes 1.02 secs per image. The run time can be approximated by adding the per image times for each stage and multiplying it by the number of focal stack images.

5.4.5 Data availability

The results presented in this Chapter and our code are available at `https://github.com/rishabhsharma27/Depth_estimation_results`.

5.5 Discussion and Conclusion

In this chapter we proposed an algorithm that uses depth from defocus to estimate the depth maps. Specifically, we divide the focal stack and all-in-focus image into smaller patches of size 4 x 4 pixels, and compared the FFTs of these patches to find the closest match to the all-in-focus image patch. We showed that using these FFTs of the patch instead of the RGB patches makes the algorithm more resilient to noise. Further, we showed that using the frequency domain to shift the sub-aperture views and the median value generates focal stacks with sub-pixel accuracy, even at small focal distances, increasing the accuracy depth precision for our depth maps near occlusions.

Our initial depth estimation estimates the depth range of the light field image. We then only generate the focal stack within that depth range, which reduces the number of redundant images, reducing the possibility of misdetections and the runtime. The gradient addition step described in Section 5.2.2 drastically improves the accuracy of the depth as shown in Fig. 5.6. In this step, we add the gradient of the image to itself. As in a focal stack image, only the region in focus will contribute to the image's gradient, enhancing the textured regions and boundaries for the in-focus regions.

The Fig. 5.17 and Fig. 5.18 in Section 5.3 shows the Mean Squared Error (MSE) for the different focal patches with the patch of the all-in-focus image when the depth is estimated correctly and when it is misdetected. In our current approach, we only look at the lowest value for the MSE and assign the depth accordingly. This approach makes the depth map accuracy more dependent on the refining step. In our future work, we intend to use a varying threshold value that would be calculated for each focal stack patch. If the number of minima below the threshold is more than one, we would further analyze that patch to ensure that the correct depth value is detected, reducing the dependence on the refining step.

We refine the depth map in two stages; one that looks at the number of pixels at a particular depth value and the second that looks for isolated patches with depths different from their neighboring patches. But as part of the refining step, we also use a global blurring filter on the entire depth map. This approach tends to cause misdetections for thin objects in the image and might get removed entirely from the depth map.

Chapter 6

Depth estimation for focal stack from 2D camera capture

6.1 Introduction

In this chapter, we extend our work from Chapter 5 to estimate the depth map using a focal stack captured by a 2D camera, rather than from a light field image. We analyze the different parameters to capture the focal stack, resulting in the most accurate depth map that can be used for light field synthesis. We also address the challenges posed by differences in the lighting conditions and magnification of the focal stack with the all-in-focus image.

6.2 Methodology

The methodology presented in this chapter uses the focal stack and all-in-focus image captured by a 2D camera to estimate the disparity map. The flow of the algorithm is represented in Fig. 6.1. As shown in Fig. 6.1, the methodology can be divided into six main sections: focal stack and all-in-focus image capture, focal stack distortion correction and pre-processing, patch generation, FFT, sharpness and gradient comparison for each

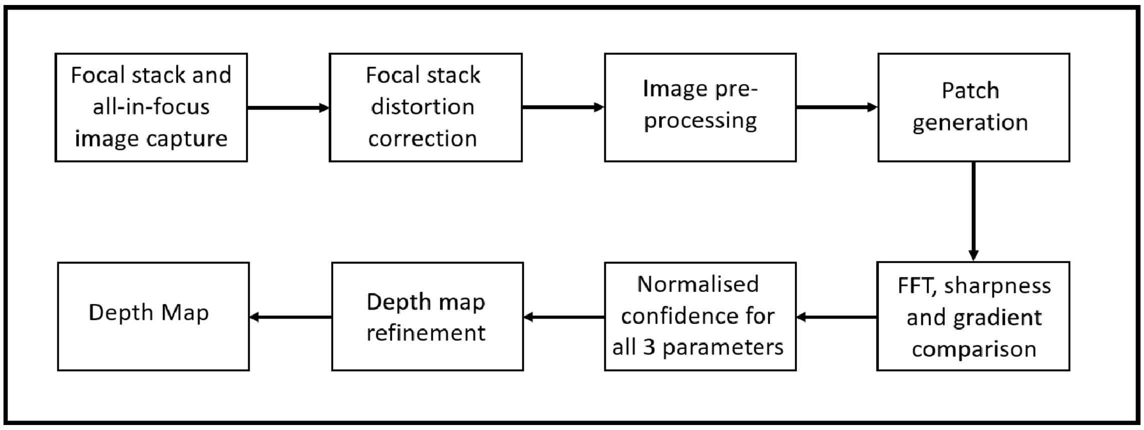

Focal stack and all-in-focus image capture
Focal stack distortion correction
Image pre-processing
Patch generation
FFT, sharpness and gradient comparison
Normalised confidence for all 3 parameters
Depth map refinement
Depth Map

FIGURE 6.1: Flow of the proposed algorithm for a focal stack captured by a 2D camera.

patch, confidence estimation for each patch, and depth refinement. The depth estimation algorithm presented in this Chapter is based on the same depth from defocus concept presented in Chapter 5.

6.2.1 Focal stack capture

Two crucial parameters must be considered to capture the focal stack: the f-stop and step size. The f-stop number governs the depth of focus for each image, while the step size moves the focal point by a set amount between two consecutive images in the focal stack. In the case of a camera lens, the step size is the amount by which the focal point between two successive images in the focal stack.

In this work, we use a Canon 600D camera with a 50mm prime lens; the f-stop for this lens can vary from f/1.8 to f/22. None of the existing focal stack datasets for 2D images contain the all-in-focus image, so we need to capture the images ourselves. Table 6.1 shows the number of images captured in the focal stack to cover the entire scene from the point closest to the camera to the furthest point according to the lens's focal length. We capture the all-in-focus image using an f-stop of f/22, as a higher f-stop has a larger depth of field, i.e., too much of the scene may be in focus for our purposes. Our initial test conducted by capturing focal stacks with varying f-stop and step size revealed that depth estimation accuracy depends on both the step size and the f-stop, which are interdependent parameters. That is, since a smaller f-stop has a smaller depth of field, the step size needs

Table 6.1: Step size vs. number of images in the focal stack

Step size	**50**	**75**	**100**	**125**	**150**
No. of images	40	27	21	17	15

to be smaller; thus, the number of images in the focal stack is larger. On the contrary, if a larger f-stop is selected, the depth of field is larger, and the step size needs to be higher to avoid overlap between the regions in focal in consecutive images in the focal stack. The f-stop and step size selection depends on the application for which the depth map is being used. Since we intend to use the depth maps for light field synthesis, for our work, the f-stop of f/1.8 and a step size of 100 produces the most accurate depth maps for our application. As we don't have the ground-truth depth map of the scene, we confirm the depth map accuracy by reconstructing the all-in-focus image and comparing the similarity index with the camera capture all-in-focus image.

6.2.2 Focal stack distortion correction and image pre-processing

As a camera changes focus on capturing the images in the focal stack, its field of view expands or contracts slightly because image magnification changes with sensor distance [98]. There can be two ways to tackle this problem: the first approach is to implement the distortion correction algorithms that uses the translation-invariant SURF algorithm [54], to scale and align the focal stack image according to the all-in-focus image. However, as the focal stack images only have part of the image region in focus, the feature can sometimes be undetected or misdetected. The second approach takes advantage of the fact that any two consecutive focal stack images are captured at a known step size: in our case, a step size of 100. This means that we can use a predefined scaling factor calculated according to the step sizes and f-stop value. The scaling also depends on the focus point, and f-stop used to capture the all-in-focus image. We capture the all-in-focus image for all the images in the same way, by choosing the focus point as the mid-point of the focal length of the lens.

We use the SURF algorithm and a predefined scaling variable in our work. To start, all the images in the focal stack are scaled according to their position in the focal stack. Then we use the SURF algorithm for two reasons, firstly to check the number of matching points in the reference all-in-focus and focal stack images. If the number of matching points is below a particular value, we remove those images from the focal stack. Secondly, to ensure that the scaled images are accurately scaled, examine the translation variable of the SURF algorithm. If the translation variable for scaling and rotation is within a particular threshold we accept the change, but if not then we remove the image from the focal stack.

The pre-processing step is the same as for the depth estimation for light field images presented in Chapter 5, section 5.2.2, where we add the gradient of the focal stack images to itself. Adding the gradient ensures that the object boundaries and textured regions are exaggerated in the focal stack images. The textured areas of the image that are in focus maximally contribute to the gradient image, while the out-of-focus objects contribute the least. This pre-processing step drastically reduces the number of misdetected patches and the dependence on the post-processing steps.

6.2.3 Patch generation

We use the same two cross patches of size 4 x 4 pixels presented in Chapter 5, section 5.2.3 and shown in Fig. 5.7. As shown in Fig. 5.7, the red and green squares are the pixels considered for matching with the all-in-focus image patch, but only the pixels highlighted in red in the red square and the pixels highlighted in green in the green square are used to generate the depth map.

6.2.4 FFT, sharpness and gradient patch comparison

The depth estimation algorithm presented in Chapter 5 only uses the FFT of the focal stack patches to estimate the depth map. However, using only the FFT of the patches for the 2D camera captured focal stack produces errors in the depth map, especially for the smooth textured regions. The results shown in Fig. 6.2(B) and (E) show the errors marked by the red box in the estimated depth map for the Gnome and Owl images, respectively. We believe that these errors are due to the focal stack being captured with an f-stop of f/1.8, whereas the all-in-focus image is captured with an f-stop of f/22. The difference in the f-stop causes the lighting conditions to change slightly, which makes using only the FFT for comparison unreliable.

An intuitive way to solve this problem would be to use color or luminance matching algorithms to match the color of the all-in-focus image with the focal stack images. The issue with the approach though is that since the focal stack has only part of the image region in focus, these algorithms produce incorrect color correction in most cases. We address this problem by using the sharpness and gradient of the patches for comparison with the FFT, where the gradient is the directional change in the pixel values in the image. It is calculated by taking the derivatives in the horizontal and vertical directions in the image. The algorithm we use to estimate the sharpness uses the gradient of the patches and then calculates the sharpness by taking the average of the sum of normals [99]. We calculate the MSE for the focal stack patches with their corresponding all-in-focus patch for all three comparison measures i.e, we use the FFT, sharpness and gradient measure.

Once we have the MSE value for the FFT, and sharpness and gradient for the patches in the focal stack, we normalize the values for each set of focal stacks, where a set is defined as the group of the same patch for each focal stack image. This normalization step is added to combine the three measures in the next step. We normalize the values for each set of focal stacks by dividing the mean of the set by the individual MSE values in the set. To compare all three measures and calculate their confidence scores, we adopt the normalisation techniques proposed by Tao *et. al* [21]. We calculate the maximum values by concatenating all three measures and then dividing each element for all three measures by that maximum value. This step normalizes the confidence score for all three measures to compare them. To estimate the depth map, we check if all three measures point to the same maxima. If it is the same maxima, we choose that value as the disparity of the patch. Otherwise, we check if two of the three measures point to the same maxima. If none of the measures points to the same maxima, we combine the three and choose the maxima for that value. The image shown in Fig. 6.2(C) and (F) show the estimated depth maps for the Gnome and Owl images using the FFT, sharpness and gradient measures,

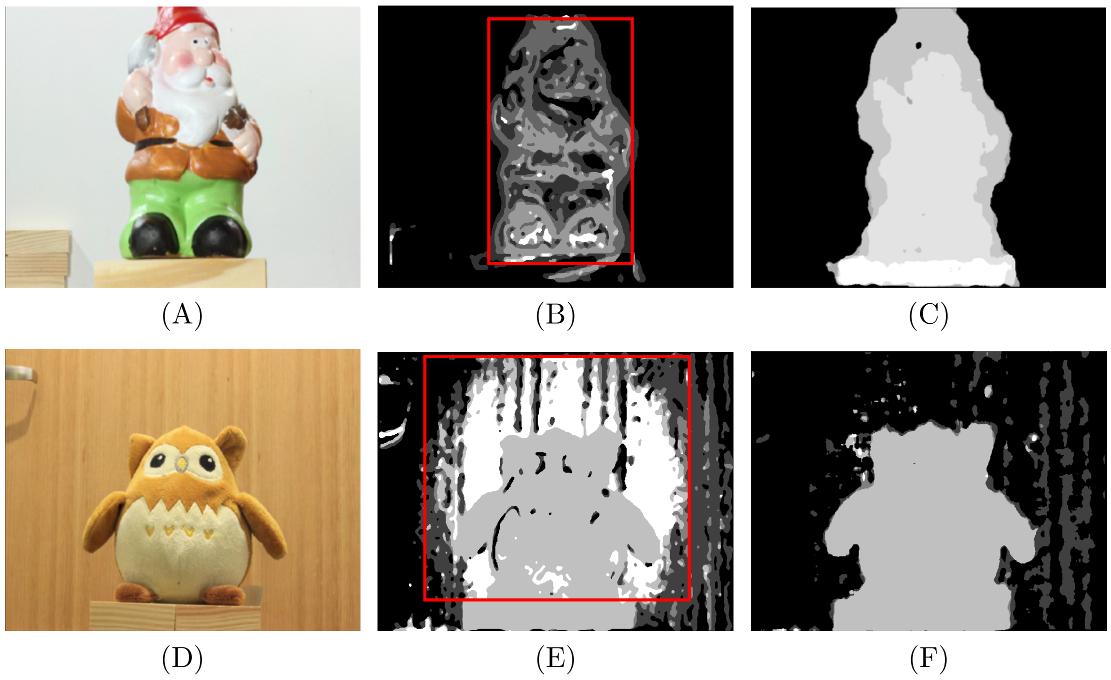


FIGURE 6.2: (A) and (D) are the all-in-focus image, (B) and (E) are the estimated depth map using only the FFT patches for comparison, and (C) and (F) are the estimated depth map using only the FFT, sharpness and gradient patches for comparison.

respectively. It is clear from the regions of the image in Fig. 6.2 marked by the red box that using the three measures improves the depth map, especially for the smooth textured areas in the image.

6.2.5 Depth refinement

The depth map evaluated up to this stage has a few patches that are misdetected, and since the patches are shaped as a cross, it creates a depth map with jagged edges. We use the same techniques described in Chapter 5, Section 5.2.4 to refine the edges and get rid of the misdetections. Fig. 6.3 shows the comparison between the estimated depth map before and after this refinement step. The disparity map is refined in two steps using an iterative approach to remove the jagged edges and correct any misdetections. Using the histogram, we first check for the number of pixels present at each depth. Those pixels that fall below the threshold are filled with the maximum likelihood values occurring most in the neighboring pixels. In the second step, we consider the patches instead of looking at individual pixels. If any isolated patches have a different depth than the adjacent patches,

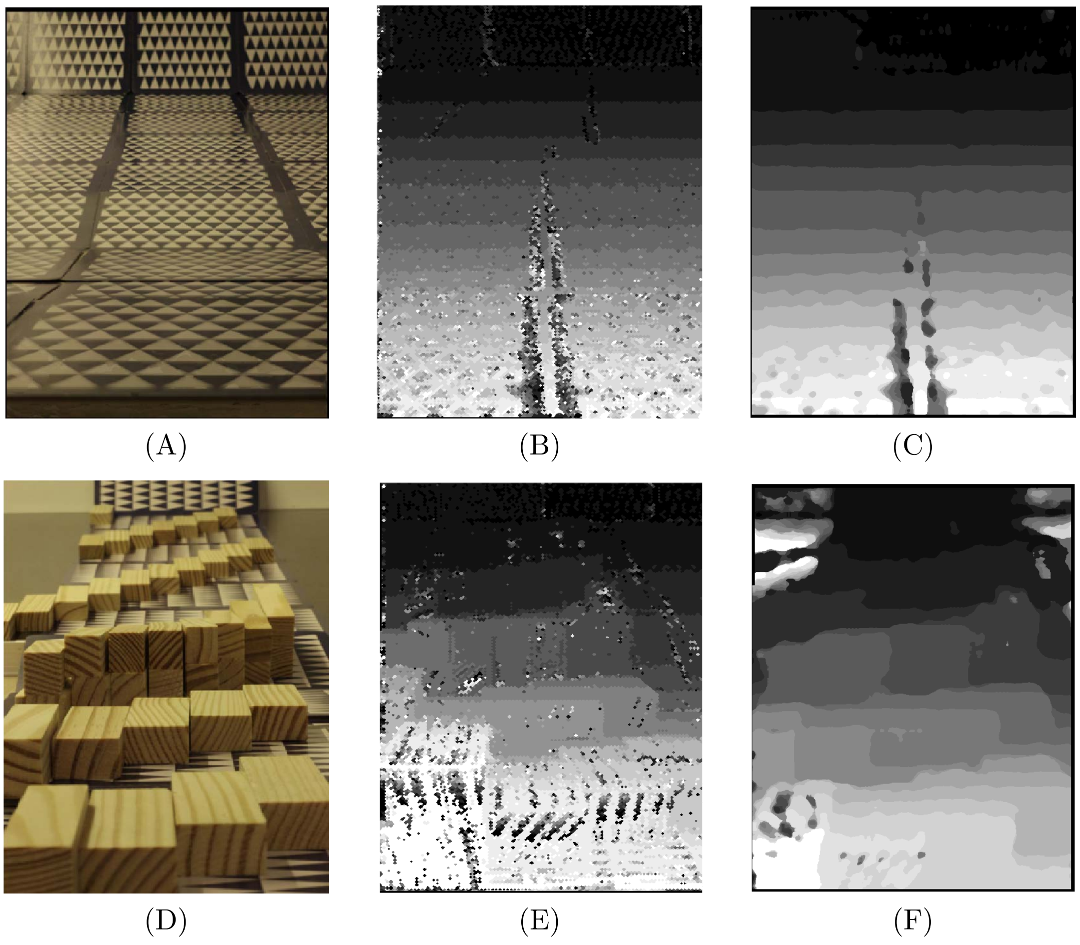


FIGURE 6.3: (A) and (D) are the all-in-focus image, (B) and (E) are the estimated depth map before refinement, and (C) and (F) are the estimated depth map after refinement.

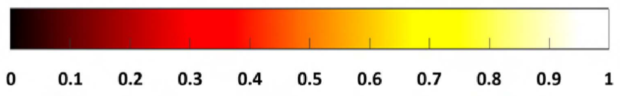


FIGURE 6.4: SSIM error map colorbar.

they are filled with the maximum likelihood values occurring most in the neighboring patches.

Table 6.2: Dataset image complexity

	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
Image	Object image-1	Object image-2	Gnome image-1	Gnome image-2	Jar image	Cards image	Blocks image	Owl image
Multiple objects	×	×	○	○	○	×	×	○
Complex patterns	×	×	○	×	○	×	×	×
Smooth texture	×	○	×	×	×	×	○	×

6.3 Experimental results

6.3.1 Dataset

To evaluate the depth map estimation approach of this Chapter, we captured the data with a Canon 600D camera with a 50mm prime lens. The data is publicly available, and the link is given in Section 6.3.3. Table 6.2 shows the composition of the different images in the dataset in terms of the number of objects, complex texture and smooth regions in the images. The crosses in Table 6.2 mark the images that fulfill the above mentioned criteria.

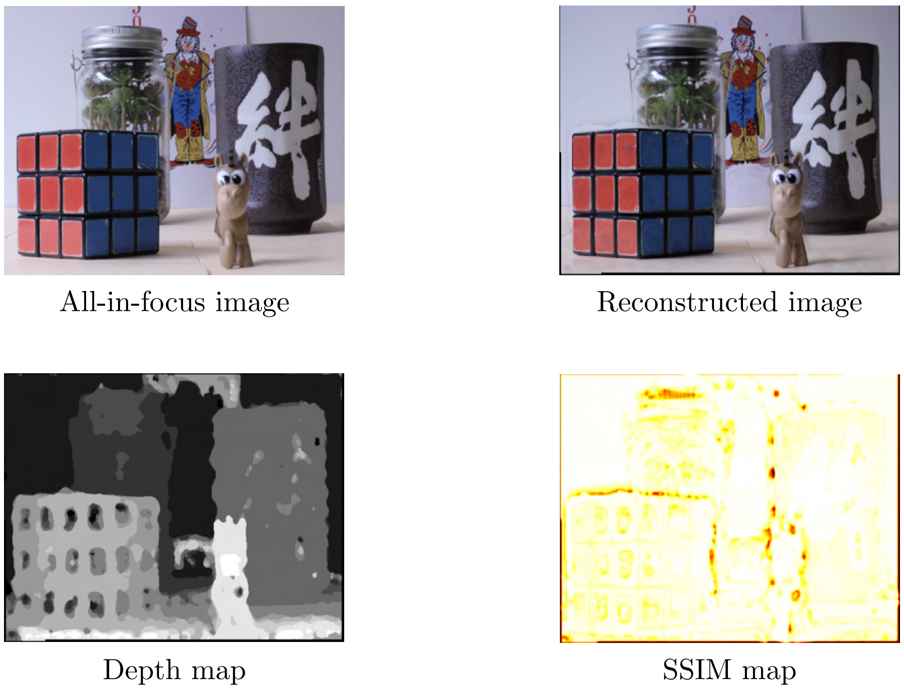
As explained in 6.2.1, we intend to use the depth maps for light field synthesis and in our work, we use an f-stop of f/1.8 and a step size of 100 for the focal stack. For the all-in-focus image, we use an f-stop of f/22, as a higher f-stop produces an image with a larger depth of field. As shown in Table 6.1, a step size of 100 corresponds to twenty-one images in the focal stack, plus one all-in-focus image. We then calculate the accuracy of the depth map by reconstructing the all-in-focus image with the depth map and focal stack. Once we have the reconstructed all-in-focus image, we use the Structural Similarity Index Measure (SSIM) to measure the similarity with the all-in-focus image captured by the camera. The pixels in the SSIM map most similar to the all-in-focus image appear white, where the similarity index is close to 1. In contrast, the regions in the image least similar appear red, where the similarity index is close to 0, as shown in Fig. 6.4.

TABLE 6.3: SSIM score for the reconstructed all-in-focus image.

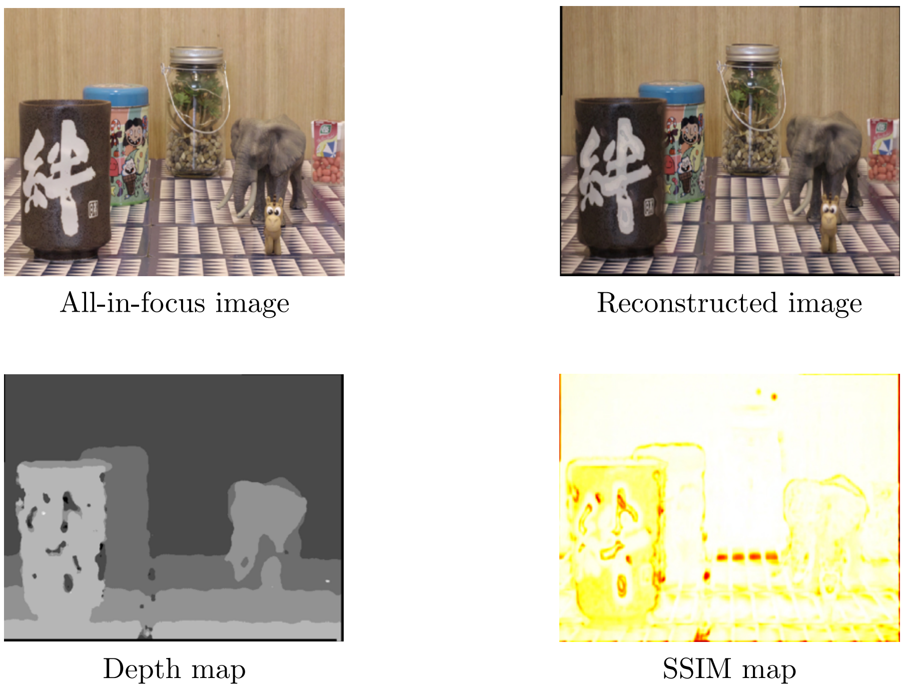
	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
Image	Object image-1	Object image-2	Gnome image-1	Gnome image-2	Jar image	Cards image	Blocks image	Owl image
SSIM score	0.94	0.92	0.8	0.96	0.95	0.96	0.94	0.85

6.3.2 SSIM Analysis

Fig. 6.5 and Table 6.3 show the visual comparison and quantitative results for the images in the dataset. It can be seen from the reconstructed images (A), (B), (C), (D), (E), and (H) in Fig. 6.5 that when the difference between the foreground and background objects is large, the boundary of the foreground objects are misdetected. The yellow color around the boundaries of the objects in the SSIM map reveals this error. Images (F) and (G) in Fig. 6.5 have a continuous depth change in the image, and there is no large difference between the foreground and background objects; thus, the boundaries are estimated accurately. The SSIM map for images (C) and (H) show that the wooden background texture is completely misdetected, and a closer inspection of the reconstructed image shows the background to be blurry. However, important to note is the background still being estimated as the same surface. This is because of the initial focal stack correction step presented in Section 6.2.2. Other than correcting the magnification for the focal stack images, we also remove the images from the focal stack that don't have any regions in focus. We do this by checking the number of SURF matching points of the focal stack with the all-in-focus image, and if it falls below a threshold, we remove the image. For foreground objects, the object boundaries become the matching points for the SURF algorithm, so even if the objects have a smooth texture, the number of matching points is above the threshold, and the image is not removed from the focal stack. On the contrary, if the background region is smooth and has no other objects, the algorithm removes the image for the focal stack. Therefore, the SSIM measure for images (C) and (H) in Table 6.3 is 0.8 and 0.85 respectively. Since images (C) and (E) have a wooden block and the card is close to the background, the focal stack focused on the background is not removed from the focal stack, and the background is estimated accurately.



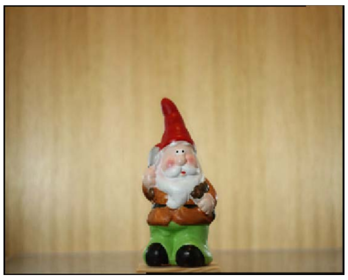
(A) Objects image-1

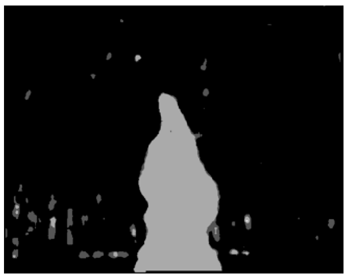


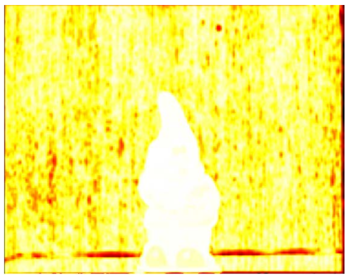
(B) Objects image-2

FIGURE 6.5: *Cont.*

All-in-focus image

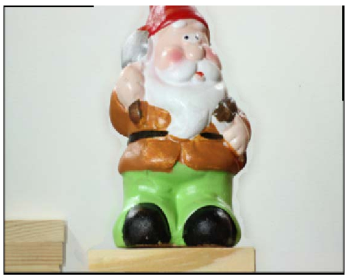

Reconstructed image

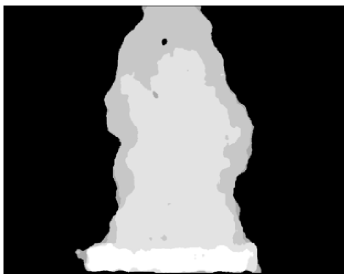

Depth map

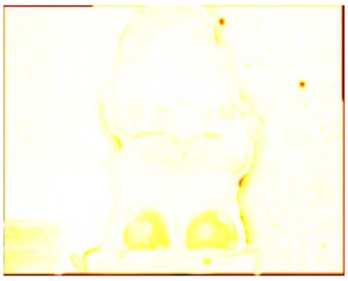

SSIM map

(C) Gnome image-1

All-in-focus image

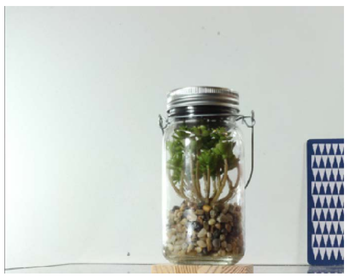

Reconstructed image

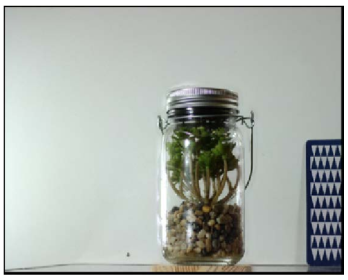

Depth map

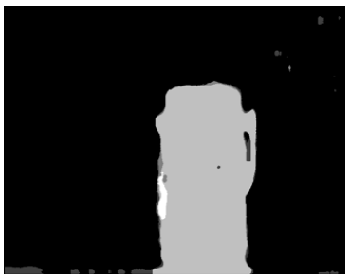

SSIM map

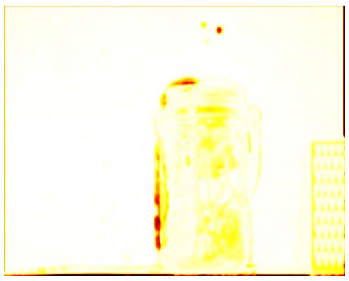
(D) Gnome image-2

FIGURE 6.5: *Cont.*

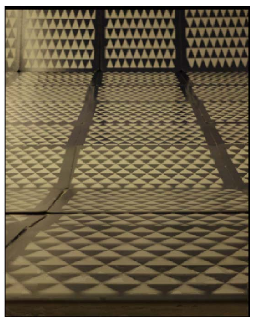

All-in-focus image

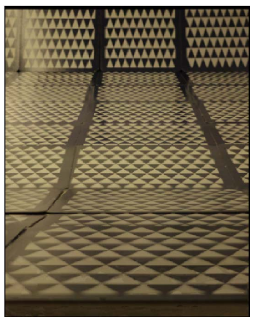

Reconstructed image

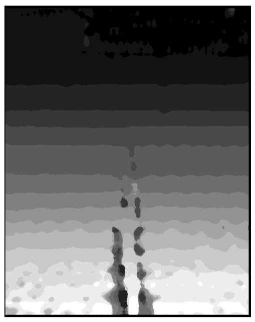

Depth map


SSIM map

(E) Jar image


All-in-focus image

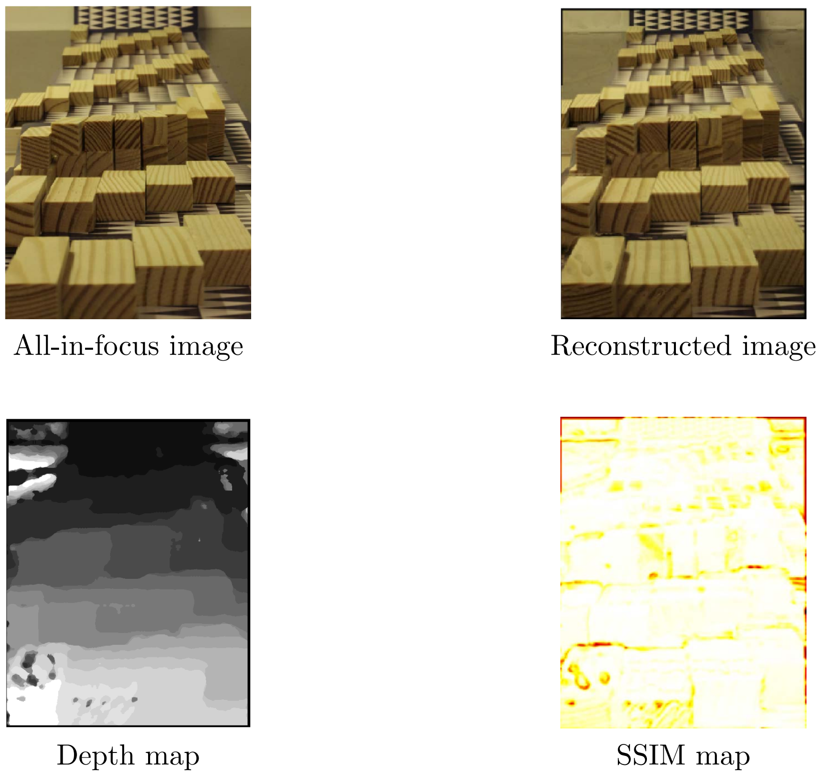

Reconstructed image


Depth map

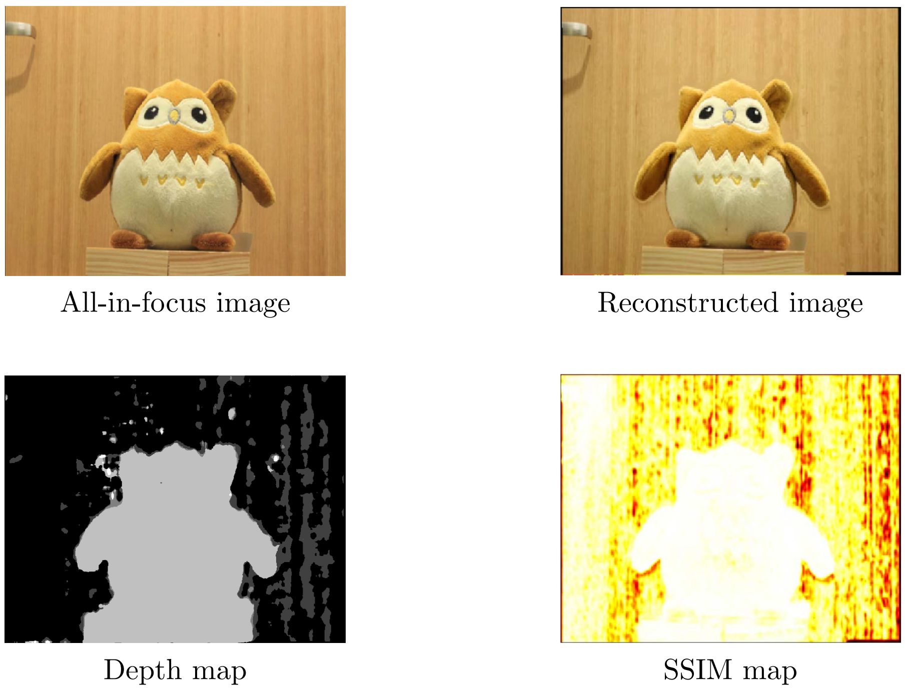
SSIM map

(F) Cards image

FIGURE 6.5: *Cont.*



(G) Blocks image



(H) Owl image

Figure 6.5: Visual comparison of the all-in-focus image and the all-in-focus reconstructed using the focal stack and depth map; the SSIM map is the similarity index between in the all-in-focus image and the reconstructed all-in-focus image.

6.3.3 Data availability

The results and data presented in this Chapter are available at `https://github.com/rishabhsharma27/2D-camera-focal-stack-Depth-map`.

6.4 Discussion and Conclusion

In this chapter we extended our depth estimation approach to using the focal stack captured by a 2D camera, focusing on light field synthesis using focal stacks. We show that capturing the focal stack with an f-stop of f/1.8 and a step size of 100 covers the scene with sufficient depth levels to synthesize a light field image using the depth map and all-in-focus image. We address the differences in lighting conditions in the all-in-focal and focal stack image by combining the FFTs of the image regions with the regions' sharpness and gradient and using their combined confidence score to estimate the depth map.

In Section 6.2.2, we rescale the focal stack images with reference to the all-in-focus image and remove the focal stack images that don't have any part of the scene in focus. Removing these images from the focal stack reduces the possibility of misdetections. For rescaling, we take advantage of the fact that the focal stacks only change in magnification. We use a predefined scaling factor that we calculate according to the f-stop value used to capture the focal stack. But if different camera parameters are used to capture the focal stack, we can use the SURF algorithm for rescaling. As mentioned above, the focal stack images change only in magnification, and if the consecutive images in the focal stack are captured with constant step size, then the magnification between consecutive images changes gradually. Using this knowledge, we can analyze the translation parameters of the SURF algorithm for each focal stack image and ensure that the scaling is done correctly.

In the next chapter, we extend our techniques from Chapter 5 and 6 of depth estimation using focal stacks to synthesize a light field image.

Chapter 7

Light field image synthesis

7.1 Introduction

This chapter proposes a light field synthesis algorithm using the focal stack and all-in-focus image. We investigate how to mimic the apparent shifting of the all-in-focus image according to the depth values to synthesize the sub-aperture views. We also analyze how we can extract information from the input data to fill the gaps generated in the synthesized sub-aperture views, to ensure high accuracy for both horizontal and vertical views.

7.2 Methodology

The methodology presented in this chapter exploits the characteristics of focal stack images and the all-in-focus image to generate a light field image with an angular resolution of 9 x 9. The 9 x 9 resolution is chosen to have the same angular resolution as the images in the dataset; thus, the accuracy of the algorithm can be calculated for the entire light field image. The flow of the algorithm is represented in Fig. 7.1. As shown in Fig. 7.1, the methodology can be divided into three main stages: depth estimation, sub-aperture view synthesis, and RGB and depth map filling for occluded regions.

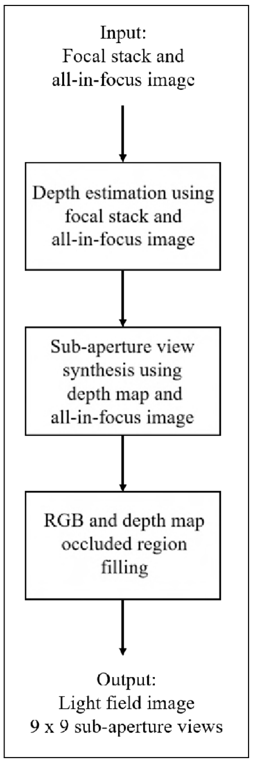


FIGURE 7.1: Light field synthesis algorithm flow

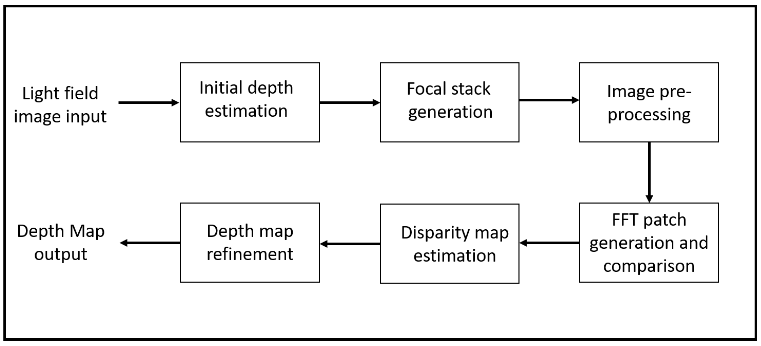


FIGURE 7.2: Depth estimation algorithm flow.

7.2.1 Depth estimation

We exploit the characteristics of focal stack images to generate a disparity map that is used to synthesise the light field image. Our algorithm uses the concept of depth from defocus by a one-to-one comparison between each focal stack image and the central all-in-focus image. This estimation approach is also noise-resilient, and outperforms the current state-of-the-art benchmark algorithms in the presence of noise [14]. Note that the below only overviews our depth estimation approach; more details are presented in our previous publication [14].

7.2.1.1 Focal stack generation and image pre-processing

Our depth estimation algorithm works with a focal stack image captured by a camera, or generated using the light field image. For the purpose of our work, we use the focal stack images generated using the light field image as we are able to validate the accuracy of the synthesized light field view by using a similarity index metric with the original light field image views.

The focal stack is produced from the light field image through the application of a shift-sum filter. This depth-selective filter operates akin to a planar focus, involving the shifting of sub-aperture images to a uniform depth and subsequently combining them to generate the 2D image. The average of the shifted sub-aperture pixels values is used for refocusing, as it replicates the blur around depth discontinuities in focal stacks captured by a camera.

To minimize the number of misdetections, the gradient of the image is added to itself to ensure that all the edges and textured regions in the image are well defined in both the central all-in-focus image and the focal stack images. The advantage of gradient addition relies on the fact that in focal stack images, the textured regions in the image that are in focus maximally contribute to the gradient image, while the out-of-focus objects contribute the least. By undergoing this pre-processing step, it guarantees an accentuation of object boundaries and textured regions in the focal stack images. Consequently, there is a significant decrease in the occurrence of misdetections, leading to a reduced reliance on subsequent post-processing steps.

7.2.1.2 Patch generation and comparison

The focal stack images from the previous stage are divided into smaller image patches, with the individual patches then compared with the corresponding patches in the all-in-focus image. For depth estimation, we use the two patches of size 4 x 4 pixels as shown in Fig. 5.7 from Chapter 5, Section 5.2.3, the squares outlined by red and green lines. The results for depth map accuracy with different window sizes showed that smaller window sizes covered the image regions and boundaries more accurately. We can also reduce the patch size to lower than 4 x 4 pixels; however, experimental tests revealed that using a patch smaller than 4 x 4 pixels does not improve the depth map accuracy and increases the computational time. Fig. 5.7 shows the two 4 x 4 image patches in the red and green squares that are considered for matching. Since we use overlapping windows, we only use the pixels highlighted in the red square and green square for the depth map estimation as shown in Fig. 5.7.

7.2.1.3 Depth estimation and refining

The estimated depth map still has a few errors, and these are refined in two steps using an iterative approach. Firstly, the histogram of the depth map is checked for the number of pixels that are present at each depth. If the number of pixels at a particular depth falls below a threshold value, those pixels are filled with the maximum likelihood value of the pixels in the depth map at that position, i.e., using the pixel value that occurs the most times in the neighbouring pixels. The second step is similar, but instead of considering individual pixels, the patches are considered. This step checks for any isolated patches in the image that have different surrounding depth patches. Once these patches are isolated, the patch is then filled with the value of pixels with maximum likelihood in the depth map at that patch position (similar to the above).

7.2.2 Sub-aperture view synthesis using FFT-shift

Epipolar images are formed by stacking the sub-aperture images in the horizontal and vertical directions, and a representative slice through this 4D block is shown in Fig. 2.2(B) and Fig. 2.2(C), respectively. The red, green and blue parallelograms show that the slope of the line reflects the depth of the pixels. The pixels that do not appear to move in-between the sub-aperture views are seen as a straight line; this is shown by the blue parallelograms in Fig. 2.2(B) that have a zero slope. The pixels shown by the green parallelograms in Fig. 2.2(C) that are closer to the camera incline to the right, and the pixels that are further away from the camera incline to the left, as shown by the red parallelograms in Fig. 2.2.

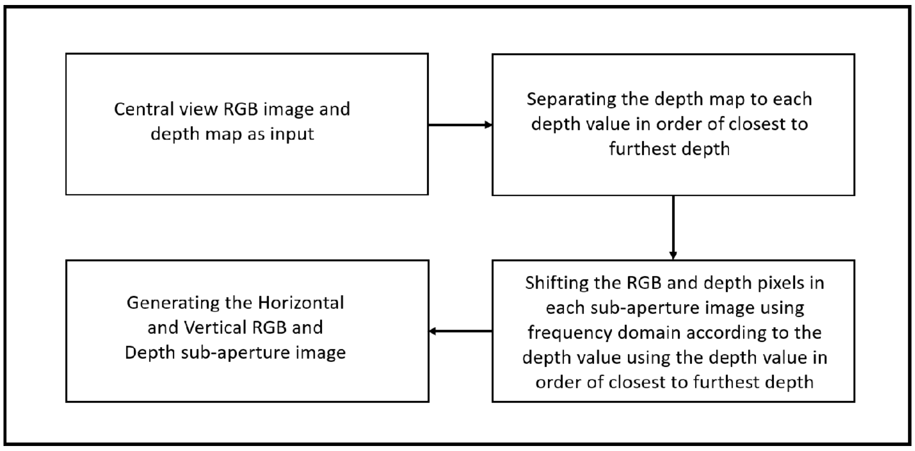
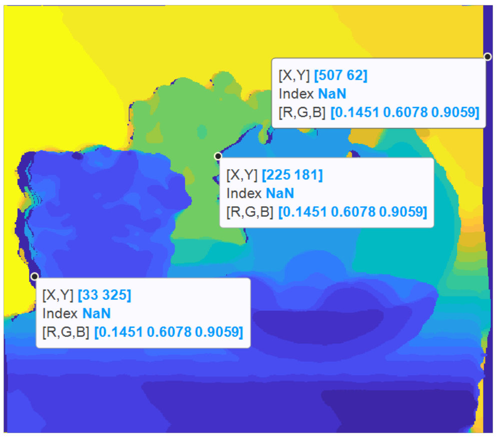

Central view RGB image and depth map as input
Separating the depth map to each depth value in order of closest to furthest depth
Shifting the RGB and depth pixels in each sub-aperture image using frequency domain according to the depth value using the depth value in order of closest to furthest depth
Generating the Horizontal and Vertical RGB and Depth sub-aperture image

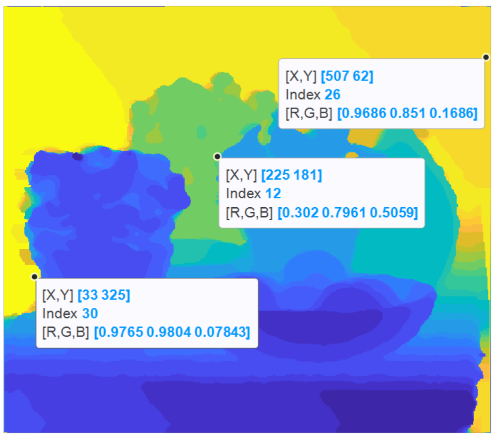
FIGURE 7.3: Algorithm flow for generating sub-aperture views

The pixels in the sub-aperture view depict the depth as they appear to be moving toward or away from the central view. In turn, if the depth is known, the pixels in all of the sub-aperture views can be filled by using pixels from the central view or the all-in-focus image.

The amount by which the pixels appear to move from the central view in the sub-aperture views is the product of the depth value and the distance of the sub-aperture view from the central view. Since the product of the depth value and the distance of the sub-aperture view from the central view can have small decimal values, we use the frequency domain to fill the sub-aperture views. Using the frequency domain to mimic the apparent shift of pixels between sub-aperture views ensures the accuracy of synthesized views at the sub-pixel level. The pixels in the sub-aperture views are thus filled from the minimum depth to the maximum depth in the depth map. As we move through different perspective views, the regions in the image closer to the camera cover the background pixels, and filling the views from the minimum depth values ensures that the regions in the image that overlap the other depths in the sub-aperture views are correctly filled.The pixels in the sub-aperture views are thus filled from the minimum depth to the maximum depth in the depth map. Fig. 7.3 shows the flow of the algorithm. Instead of filling the EPI with the shifted pixels, we directly add the RGB and depth pixels to the sub-aperture views by moving the pixels by the amount calculated by the depth value of those pixels in the depth map and the position of the sub-aperture view with respect to its position from the central view.



(A)



(B)

FIGURE 7.4: (A) Gaps generated in the depth image due to occlusion, and (B) Depth map after the gaps are filled.

7.2.3 RGB and depth map filling of occluded regions

Once we fill the pixels in the separate sub-aperture views using the all-in-focus image and depth maps, due to the difference in the depth between different regions, some parts of the image that are not visible in the central view are exposed. This also occurs in the perspective depth map views as shown in Fig. 7.4.

7.2.3.1 Depth map filling of occluded regions

In a depth map, if there are two regions at different depths and a gap is thus created, the region will always be filled by the depth value which is farther away, as the apparent movement of the foreground objects is more than the background objects between sub-aperture views as shown in Fig. 7.4(B).

7.2.3.2 Filling occluded RGB regions using the focal stack

Filling the occluded regions in the RGB images is more complex than filling the occluded regions in the depth map, as the depth map values have only two possible values to choose from: the foreground or the background depth values. In our approach, the depth values and the focal stack images are used to estimate the pixel values for the gap generated

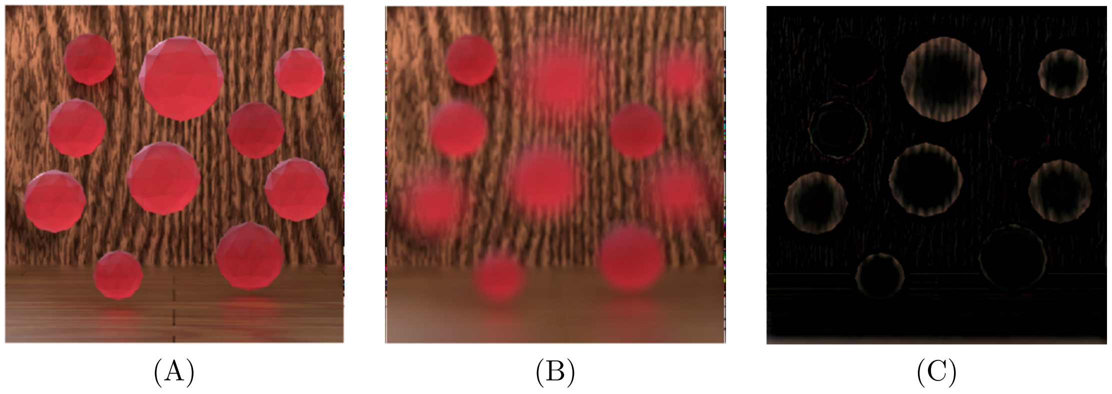


Figure 7.5: (A) The all-in-focus image, (B) the image in the focal stack focused at the background, and (C) background region minus the effect of the foreground object blur.

near depth discontinuities. Due to the defocus blur in focal stack images, when focusing on the background, parts of the background are revealed that are not visible in the all-in-focus image. The amount of blur is also dependent on the depth difference between the foreground and background object. As the pixels in the sub-aperture images also move in accordance to their depth values, the focal stack image reveals the exact amount of information required to fill the gaps, as shown in Fig. 7.5. Fig. 7.5(C) is obtained by blurring the foreground objects by the amount equivalent to the depth difference between the object and subtracting it from the image focused on the background. But since the induced blur can only approximate the lens blur, the extracted image region still contains the color tone of the foreground region.

7.2.3.3 RGB image refinement

Fig. 7.6 represents a light field image with an angular resolution of 9 x 9 views. The blue square represents the starting point for the proposed light field synthesis algorithm. We use the all-in-focus image and the estimated depth map to synthesize the central horizontal and vertical sub-aperture views; which in this depth map is represented by the green and yellow squares in Fig. 7.6. Each of the generated central horizontal views and its corresponding depth map are then used to synthesize the sub-aperture views above and below the green squares, while each of the generated central vertical views and its corresponding depth map is then used to synthesize the sub-aperture views to the right and left of the yellow squares. All of the orange squares thus are synthesized using the

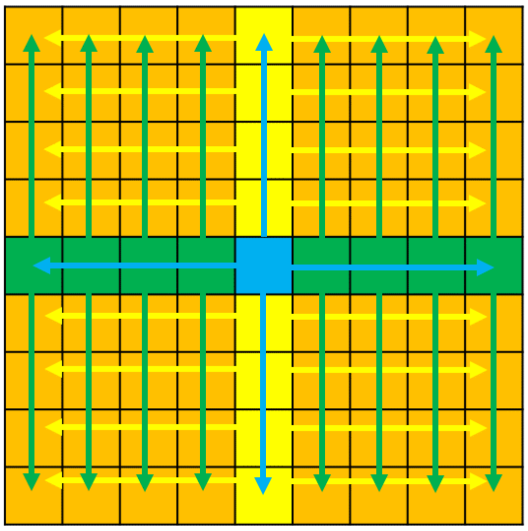


FIGURE 7.6: Showing the order in which the sub-aperture images are generated.

central horizontal and vertical sub-aperture views, and its depth map is represented by the green and yellow squares. Both the sub-aperture views generated from the horizontal and vertical views are then averaged as the final light field image.

7.3 Results

7.3.1 Dataset

The results of the proposed algorithm were evaluated on a synthetic 4D light field image dataset [3]. The dataset is widely used to validate depth estimation and reconstruction/synthesis algorithms for light field images as it contains ground-truth disparity and depth maps. The depth range for the synthetic data lies within the range of -4 to +4, and the number of focal stack images can be increased or decreased by reducing or increasing the focus interval between consecutive focal stack images.

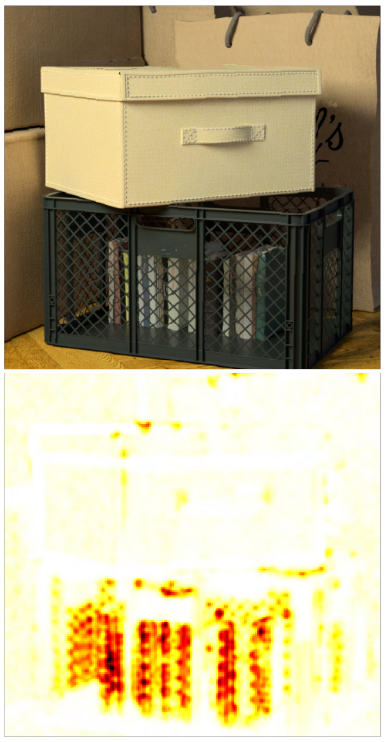
The proposed algorithm is compared to three benchmark techniques from Kalantari *et al.* [10], Chao *et al.* [11], and Zhang *et al.* [9]. We have chosen these three techniques because they are state-of-the-art for light field synthesis, and each uses a different approach. Zhang *et al.* [9] use a micro stereo pair, Chao *et al.* [11] use a stereo pair with a large baseline, while Kalantari *et al.* [10] use the four corner sub-aperture views for light field synthesis. The Structural Similarity Index Measure (SSIM) and Peak-Signal-to-Noise-Ratio (PSNR) metrics were used for evaluation.

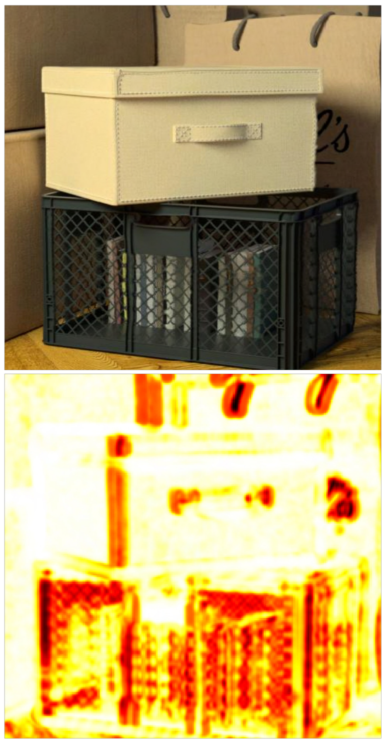
For Kalantari *et al.* [10], we utilised the trained network used by the authors to synthesize the light field images. For Chao *et al.* [11], we trained the network using the code provided by the authors. For the SSIM metric, we compare the top-left sub-aperture views for the algorithms that synthesize a 9x9 or 8x8 light field image. We can use any of the four corner images for evaluation: we use the corner views for evaluation as they show the maximum parallax from the central view. For the algorithm that only synthesizes the horizontal sub-aperture views, we use the central left-most horizontal view for evaluation. For the PSNR metric, we convert the sub-aperture light field image to the lenslet view and calculate the PSNR.

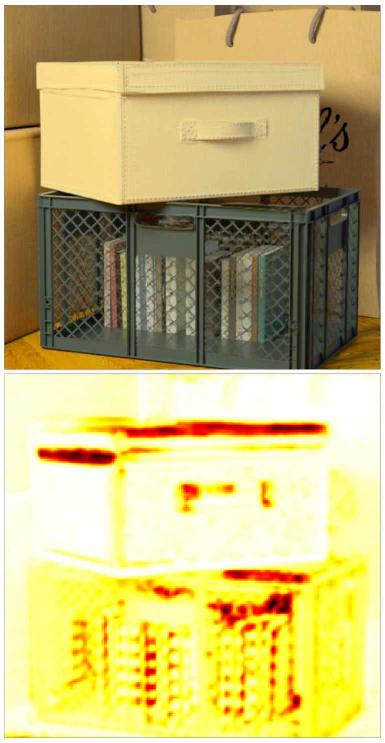
Kalantari *et al.* [10] synthesize the light field image using the 4 corner sub-aperture images with an angular resolution of 8 x 8, whereas we synthesize the light field image with an angular resolution of 9 x 9. We thus evaluate the results for comparison by using only the inner-most 8 x 8 sub-aperture views. Chao *et al.* [11] use 2 central horizontal corner sub-aperture images with an angular resolution of 9 x 9. As they train their network on 20 images from the dataset, only 4 light field images remain for testing, so we evaluate the average PSNR and SSIM for the 4 test images. The algorithm proposed by Zhang *et al.* [9] only synthesizes horizontal sub-aperture views using 2 micro baseline stereo pairs, so we evaluate the results for only the horizontal views.

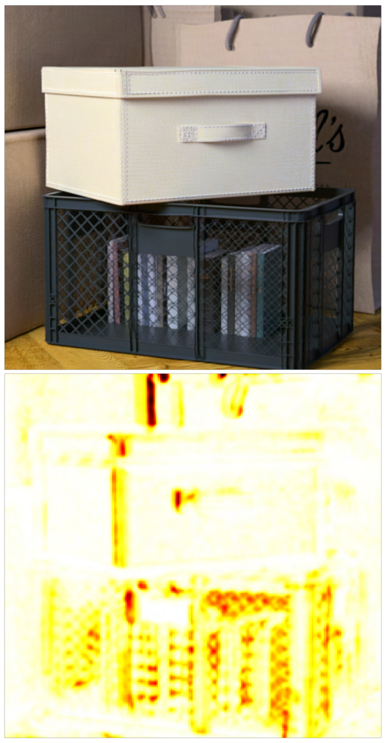
The images in Fig. 7.7 show the synthesized left most horizontal view and the SSIM error map for three dataset images. The pixels in the SSIM map most similar to the ground-truth sub-aperture view appear white, where the similarity index is close to 1. In contrast, the regions in the image least similar appear red, where the similarity index is close to 0, as shown in Fig. 7.8.

For depth estimation we use a depth from defocus technique that uses the focal stack images and an all-in-focus image to estimate the depth map. As the number of images in the focal stack govern the resolution of the depth map, the accuracy of the synthesized light field images reduces as the number of focal stack images reduces. Table 7.1 shows the average PSNR and SSIM for the proposed algorithm for different numbers of focal stack


Our result

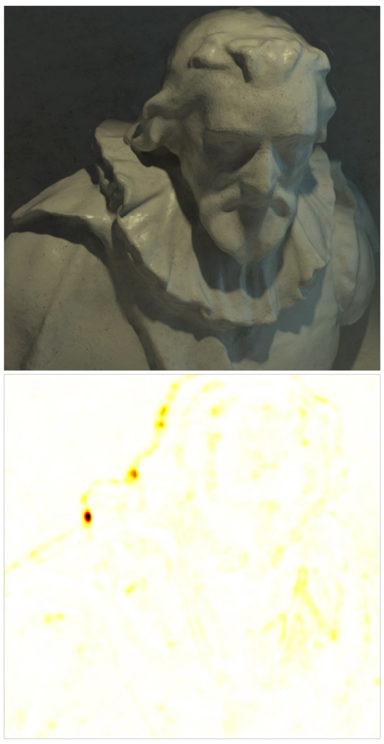

Zhang *et al.* [9]

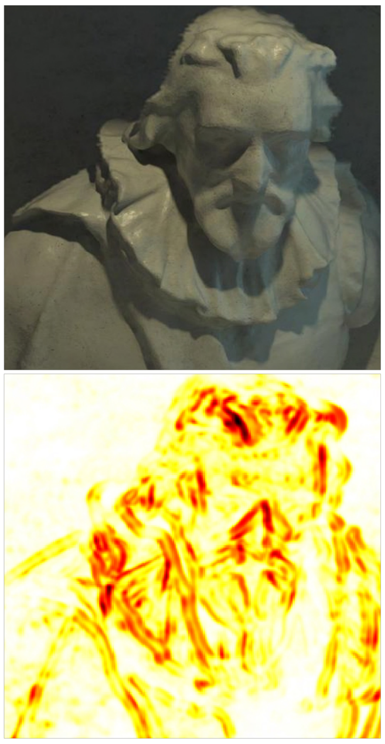

Kalantari *et al.*[10]

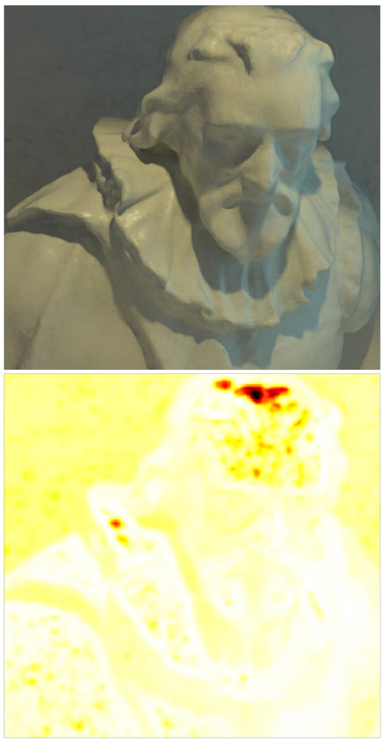

Chao *et al.*[11]

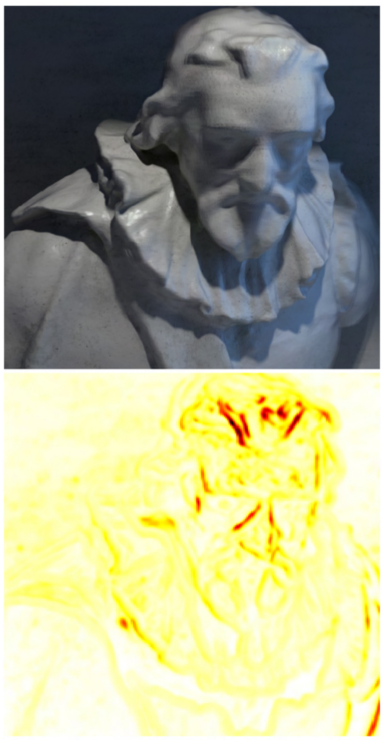
(A) 'Boxes' image

FIGURE 7.7: *Cont.*


Our result

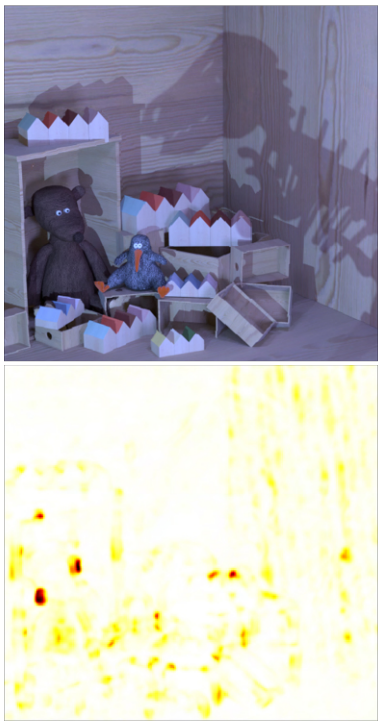

Zhang *et al.*[9]

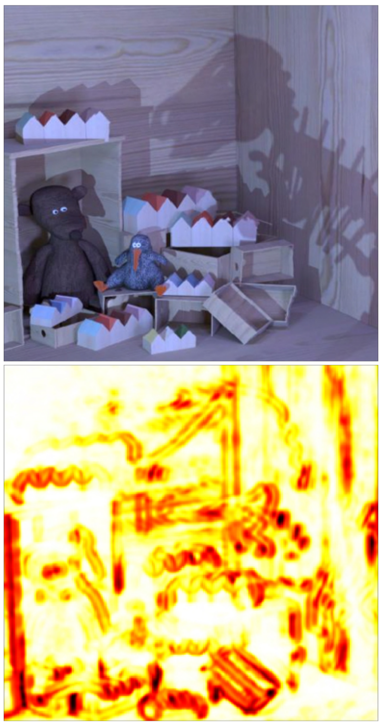

Kalantari *et al.*[10]

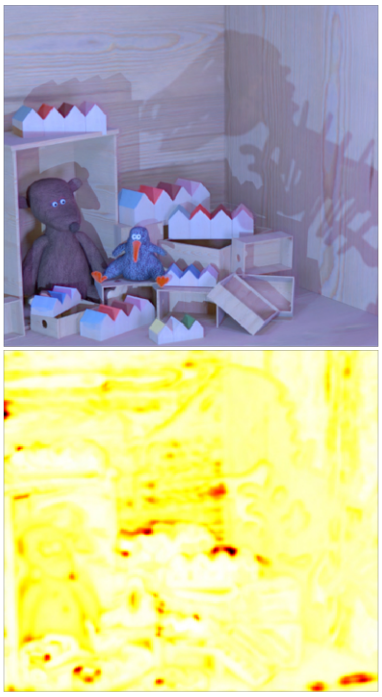

Chao *et al.*[11]

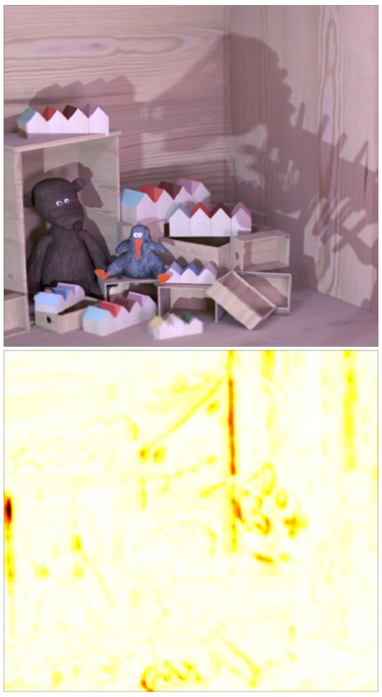
(B) ‘Cotton’ image

FIGURE 7.7: *Cont.*


Our result


Zhang *et al.*[9]


Kalantari *et al.*[10]


Chao *et al.*[11]

(C) ‘Dino’ image

FIGURE 7.7: Visual comparison for the ‘Boxes’, ‘Cotton’, and ‘Dino’ images synthesized leftmost horizontal sub-aperture view and the SSIM with the ground-truth sub-aperture view for the proposed algorithm, Zhang *et al.* [9], Kalantari *et al.* [10] and Chao *et al.* [11].

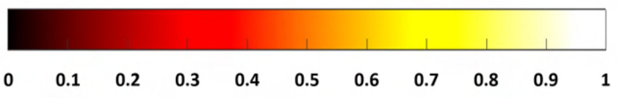


FIGURE 7.8: SSIM error map colorbar.

TABLE 7.1: Average PSNR and SSIM for all images in the dataset using focal stacks of varying sizes. For focal stack with 5 images, the focal stack images used are captured between the maximum and minimum depth value for the depth range of each image.

	41	**21**	**17**	**5***
PSNR	33.69	32.23	31.5	29.63
SSIM	0.9588	0.9461	0.9395	0.9052

images used for light field synthesis for all images in the dataset [3]. It should be noted that for the sake of generalization, the focal stack images are captured over the entire depth range for the synthetic data; that is, from -4 to +4 irrespective of the depth range of individual images in the dataset. So even though the number of focal stack images for light field synthesis are 41, 21, and 17 in Table 1, the focal stack images that have regions in focus are less than the total number of images in the focal stack. We chose the number of focal stack images as 41, 21, and 17 as these values generate consecutive focal stack images at a depth difference of 0.2, 0.4, and 0.5, respectively, for the depth range of -4 to +4. Furthermore, only the images that have regions in focus contribute to the estimation of the depth map. For a focal stack with 5 images marked with a '*' in Table 7.1, the focal stack images are captured between the maximum and minimum depth values for the depth range of each image.

7.3.2 Quantitative Analysis

The quantitative results are divided into two parts, as the method of Zhang *et al.* [9] only generates the central horizontal sub-aperture views. Fig. 7.7 and Table 7.2 show the visual comparison and quantitative results for the left most horizontal view for the Boxes, Cotton and Dino image for Zhang *et al.* [9], Kalantari *et al.* [10] and Chao *et al.* [11] using the PSNR and SSIM metrics. Table 7.3 shows the average PSNR and SSIM for Zhang *et al.* [9] and the proposed algorithm for the central horizontal views for all the images in

Table 7.2: Quantitative comparison for leftmost central horizontal synthesized view for images shown in Fig. 7.7.

	Boxes	Cotton	Dino
		Our result	
PSNR	28.41	44.99	38.42
SSIM	0.9313	0.9953	0.9901
		Zhang *et al.* [9]	
PSNR	21.50	25.72	23.15
SSIM	0.6362	0.7390	0.6801
		Kalantari *et al.* [10]	
PSNR	19.74	17.21	19.16
SSIM	0.8139	0.8902	0.9207
		Chao *et al.* [11]	
PSNR	27.34	24.85	19.67
SSIM	0.9260	0.9357	0.9355

the dataset [3]. Table 7.4 shows the average PSNR and SSIM for Kalantari *et al.* [10] and the proposed algorithm for 8 x 8 views for all the images in the dataset [3]. For Chao *et al.* [11], since 20 images are used for training the network, Table 7.5 shows the average PSNR and SSIM for the 4 test images in the dataset [3].

7.3.2.1 Quantitative analysis for central leftmost horizontal sub-aperture view

The 'Boxes' image in Fig. 7.7 consists of a crate with books in the foreground and bags in the background. None of the algorithms can accurately synthesize the fine criss-cross pattern on the crate, as shown by the yellow and red regions in the SSIM error map of Fig. 7.7(A). While our algorithm can still maintain the criss-cross pattern of the crate, the results for Zhang *et al.* [9] show distortion for this foreground pattern. For Kalantari *et al.* [10], the criss-cross pattern is invisible in some regions in the synthesized view, whereas for Chao *et al.* [11], the pattern appears to be doubled. The results for Zhang *et al.* [9] also show distortion around the edges of the crate and the box placed on the crate. For the view synthesized by Kalantari *et al.* [10], the thread pattern on the box placed on the crate is synthesized inaccurately, and the pattern appears twice in some regions. For

Chao *et al.* [11], the boundaries of the bags in the background appear to be shifted and superimposed on the image, making it appear blurred.

The ‘Cotton’ image in Fig. 7.7(B) is relatively simple as it only consists of a statue and a plain colored wall in the background. Our results show that, except at depth discontinuities, our algorithm can synthesize the image accurately for all the other regions in the image: for PSNR and SSIM in Table 7.2, the accuracy is 44.99 and 0.9953, respectively. For Zhang *et al.* [9] the synthesized image and SSIM error map show that near the head of the statue on the left side, the boundaries are pixelated, and on the right side, part of the head is stretched. It is also be seen in the SSIM error map in Fig. 7.7(B) that most of the depth discontinuities within the statue and the shadow regions in the figure are also incorrectly synthesized. For Kalantari *et al.* [10], we can see from Fig. 7.7(B) that the area within the statue is synthesized accurately, except for the regions where a shadow is cast on the statue. Some texture near the top of the head is missing, represented by the red area in the SSIM error map. The image background region is synthesized inaccurately, shown by the yellow areas in the SSIM error map. For Chao *et al.* [11], even though the SSIM score is 0.9357, the structure of the eyes, nose, and hair appear to be perceptually shifted and superimposed on the image, making it appear blurred near those regions in the figure.

The ’Dino’ image in Fig. 7.7(C) is relatively complex as it consists of a textured wooden background, the cast of a dinosaur shadow on the wall, and wooden toys and boxes. Our results show a similar trend as the ’Boxes’ and ’Cotton’ images, where our algorithm can synthesize the image accurately for all the regions in the image except at depth discontinuities: the PSNR and SSIM results show the accuracy to be 38.42 and 0.9901, respectively, as shown in Table 7.2. For Zhang *et al.* [9] the resultant image and SSIM error map show that most of the depth discontinuities in the figure are also incorrectly synthesized. It can also be seen in the synthesized view in Fig. 7.7(C) that the open wooden shelves on the left and the wooden boxes on the right have jagged edges. For Kalantari *et al.* [10], we can see that the synthesized view from Fig. 7.7(C) has no visual errors. Still, the SSIM error map shows that most regions in the image appear yellow, implying that the accuracy of the synthesized views compared to the actual light field view is between 80-90% (as indicated by the color bar in Fig. 7.8). For Chao *et al.* [11], we see a similar trend as the ’Boxes’ and 'Cotton' images, where the synthesized view has regions in the image that appear to be shifted and superimposed on the image, making it appear blurred near those regions in the figure. This effect is visible near the dinosaur shadow on the wall, and the wooden toys near the bottom of the image, where the image appears blurred.

7.3.2.2 Quantitative analysis for top-leftmost sub-aperture view

Since Kalantari *et al.* [10] and Chao *et al.* [11] synthesise the light field image with angular resolution on 8 x 8 and 9 x 9 respectively, we can compare the appropriate corner sub-aperture view to measure the algorithm's accuracy. Fig. 7.9, and Table 7.6 show the visual comparison and quantitative results for the top left sub-aperture view. The results for our algorithm for the top corner sub-aperture views reduces in accuracy compared to the horizontal views. As our algorithm uses depth maps to synthesize the sub-aperture views, the error in the depth map for horizontal views is amplified for the top and bottom sub-aperture views, which reduces the synthesized image accuracy for the top and bottom views. For all images in the dataset, the accuracy is reduced for the top left view compared to the horizontal view for our algorithm: the PSNR reduces from 33.55 to 31.24 and the SSIM reduces from 0.9713 to 0.9525. For Chao *et al.* [11], the accuracy for the top corner sub-aperture views also reduces compared to their synthesized horizontal views, but the drop in accuracy is quite significant for SSIM. For the four test images evaluated, the accuracy for the top left view compared to the horizontal view in terms of PSNR reduces from 23.74 to 21.8, whereas the SSIM reduces from 0.9093 to 0.7902. In contrast, for Kalantari *et al.* [10], as the input images used are the four corner sub-aperture views, the accuracy of the synthesized views reduces as we move towards the central views from the four corner views. For all images in the dataset, the average reduction in accuracy for the horizontal view compared to the top-left view is a PSNR reduction from 19.21 to 18.62 and SSIM reduction from 0.8872 to 0.8071.

For the 'Boxes' image in Fig. 7.9(A), as in the case of horizontal views, the top left sub-aperture view also struggles with the fine criss-cross pattern on the crate, as seen in the yellow and red regions in the SSIM error map. For our algorithm, the depth map inaccuracies cause some regions to be synthesized incorrectly. This effect is visible above the box near the upper left part of the image, where the boundaries of the bags are shifted slightly to the right. The results for Kalantari *et al.* [10] and Chao *et al.* [11] show errors in similar regions as seen in Fig. 7.9(A), which is near the top edge of the box and the crate.

For the 'Cotton' image in Fig. 7.9(B), our results show similar high accuracy for the top left view as the horizontal view, where the algorithm can synthesize the image accurately for all the regions except at depth discontinuities. The PSNR and SSIM results shown in Table 7.6 show the accuracy to be 43.07 and 0.9925, respectively. For Kalantari *et al.* [10], we can see from Fig. 7.9(B) that the background region in the image is synthesized inaccurately, as shown by the orange regions in the SSIM error map. Areas in the image with shadows near the neck and shoulder are also inaccurately synthesized. For Chao *et al.* [11], similar to the horizontal view, the structure of the eyes, nose, and hair appear to be shifted and superimposed on the image, making it appear blurred near those regions in the figure.

In the 'Dino' image in Fig. 7.9(C), our algorithm accurately synthesizes the image for all the regions except at depth discontinuities, where the SSIM error map appears yellow. For Kalantari *et al.* [10], we can see that the synthesized view from Fig. 7.9(C) has no visual errors with an SSIM of 0.9420, but this high accuracy is also because the corner sub-aperture views are used as input images for the synthesis. Still, the SSIM error map shows that most regions in the image appear yellow, implying that the accuracy of the synthesized views compared to the actual light field view is between 80-90% (see color bar in Fig. 7.8). For Chao *et al.* [11], near the dinosaur shadow on the wall and the wooden toys near the bottom of the image, parts of the image appear to be shifted and superimposed on the image, making it appear blurred near those regions in the figure.

In the 'Sideboard' image in Fig. 7.9(D), our synthesized view shows errors in two regions in the image. As our depth estimation algorithm cannot distinguish the depth for thin objects, the ceiling wires on which the lights hang from are incorrectly synthesized. The other error is near the bottom of the image, where the legs of the sideboard appear distorted. Again, this is because the depth estimation algorithm misdetected the depth of the legs. For Kalantari *et al.* [10], we can see the sideboard and the objects placed on

Table 7.3: For comparison with Zhang *et al.* [9], we evaluate the average PSNR and SSIM for only horizontal views for all images in the dataset as the algorithm only synthesizes horizontal views.

	Our result	**Zhang *et al.* [9]**
PSNR	32.23	21.1
SSIM	0.9313	0.7592

Table 7.4: For comparison with Kalantari *et al.* [10], we evaluate the average PSNR and SSIM for all images in the dataset.

	Our result	**Kalantari *et al.* [10]**
PSNR	33.69	18.6
SSIM	0.9588	0.8834

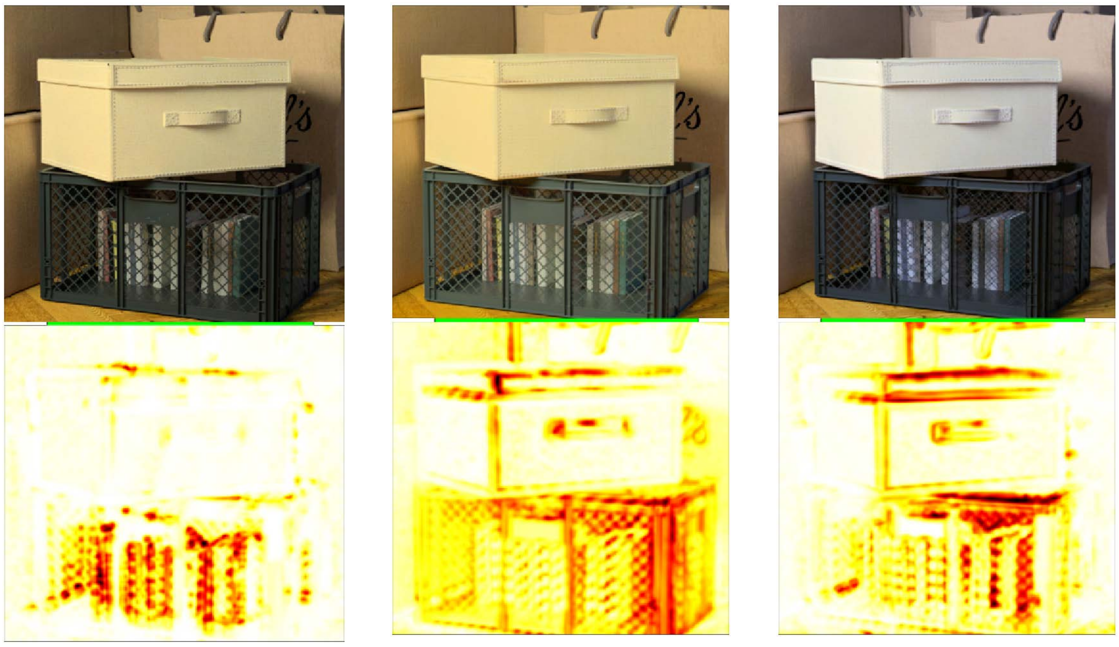
Table 7.5: For comparison with Chao *et al.* [11], we evaluate the average PSNR and SSIM for 4 test images in the dataset, as the remaining 20 images are used to train their network.

	Our result	**Chao *et al.* [11]**
PSNR	38.08	20.02
SSIM	0.9672	0.8901

the sideboard are accurately synthesized as these regions in the SSIM error map appear white, whereas all other regions appear yellow. For Chao *et al.* [11], the pattern of the wall and the objects placed on the sideboard appear to be blurred, which is again because it appears as a shifted image superimposed on the image.

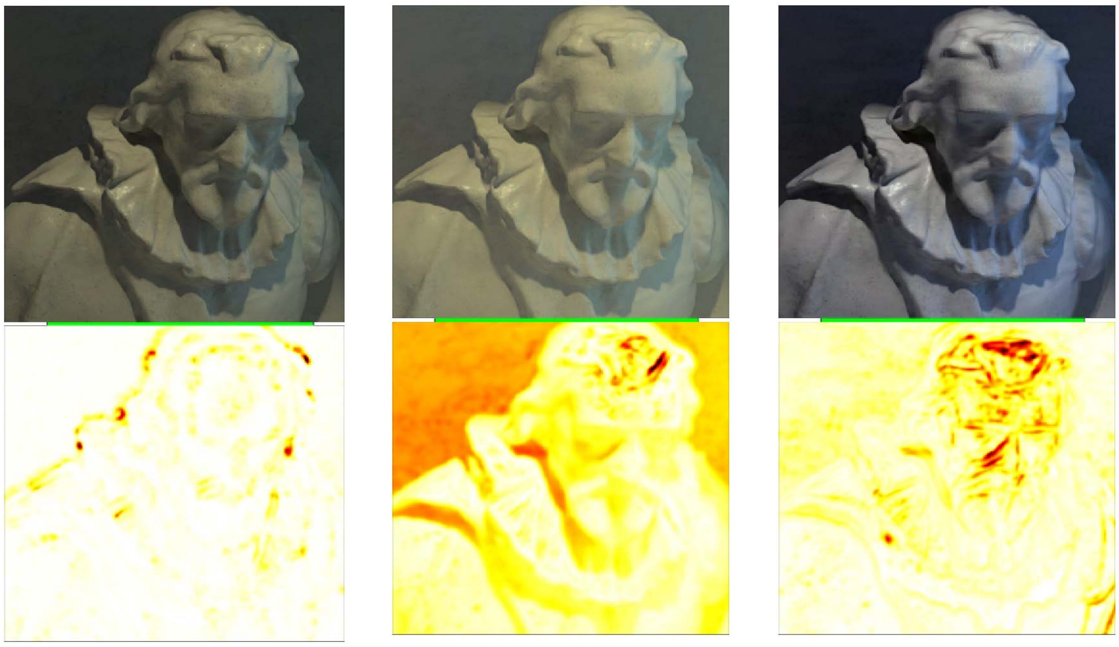
7.3.3 Quantitative analysis for real light field image

To evaluate the accuracy of our algorithm on real light field images, we use the 30-scene dataset [10]. We compare the accuracy of our algorithm with Wu *et al.* [12], Yeung *et al.* [13] and Kalantari *et al.* [10]. Table 7.7 shows the PSNR and SSIM results averaged over all 30 images in the dataset. We synthesize 7 x 7 sub-aperture views for the real light field images using the focal stack images and the central all-in-focus image. The results for Wu *et al.* [12], Yeung *et al.* [13] and Kalantari *et al.* [10] were extracted from the results presented by these authors in their respective publications. For the real images,



Our result Kalantari *et al.*[10] Chao *et al.*[11]

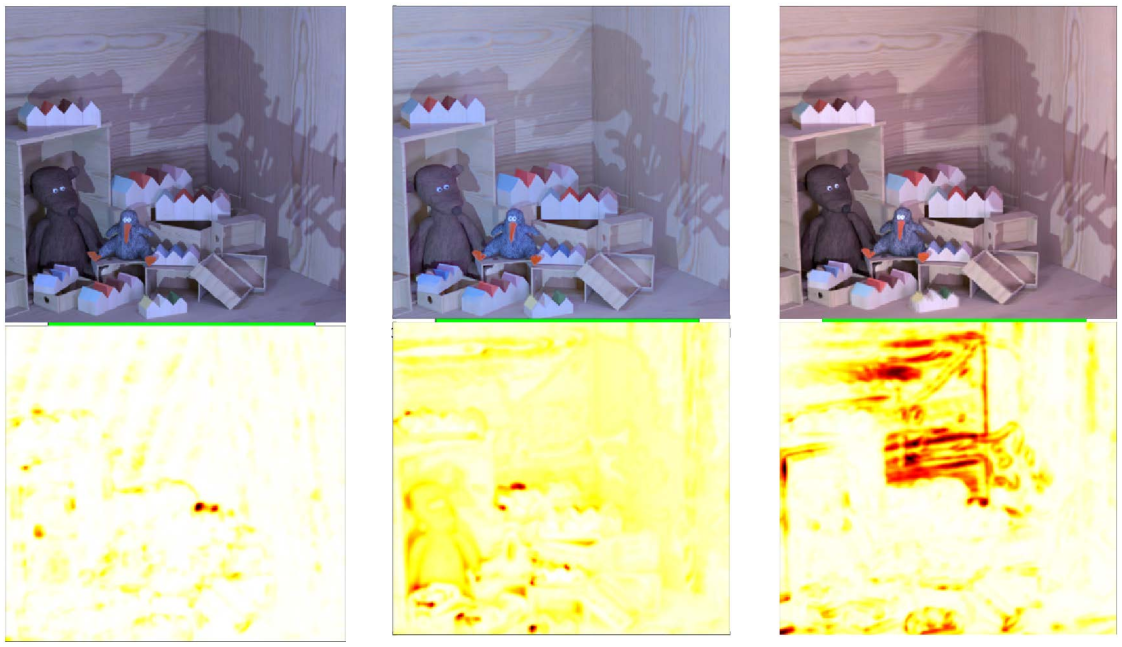
(A) 'Boxes' image



Our result Kalantari *et al.*[10] Chao *et al.*[11]

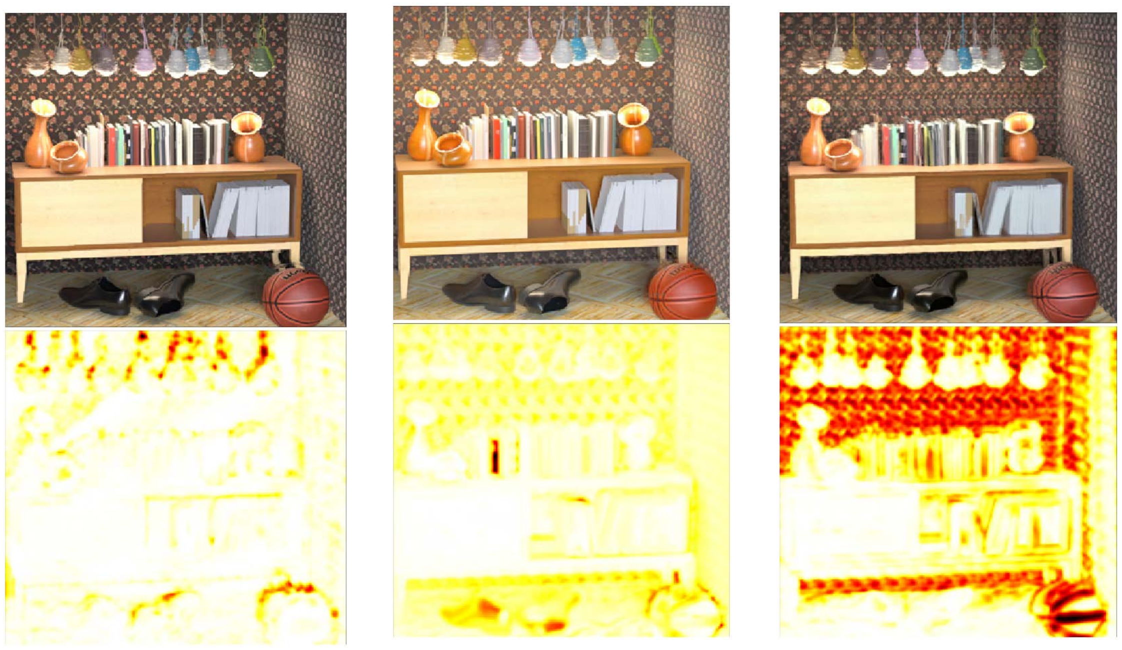
(B) 'Cotton' image

FIGURE 7.9: *Cont.*



Our result (Kalantari *et al.*[10] Chao *et al.*[11]

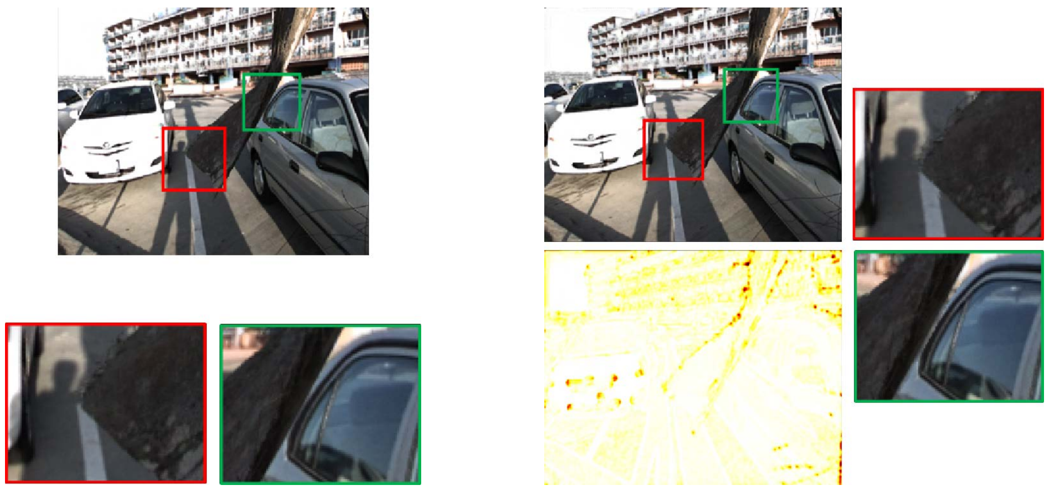
(C) 'Dino' image



Our result Kalantari *et al.*[10] Chao *et al.*[11]

(D) 'Sideboard' image

FIGURE 7.9: Visual comparison for the 'Boxes', 'Cotton', 'Dino', and 'Sideboard' images synthesized top-left sub-aperture view and the SSIM with the ground-truth sub-aperture view for the proposed algorithm, Kalantari *et al.* [10] and Chao *et al.* [11].


(a) Car

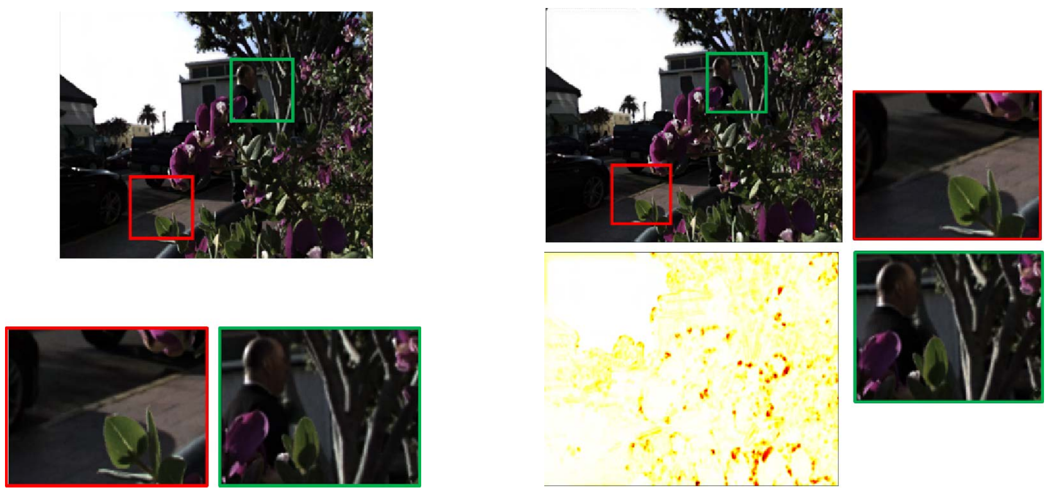
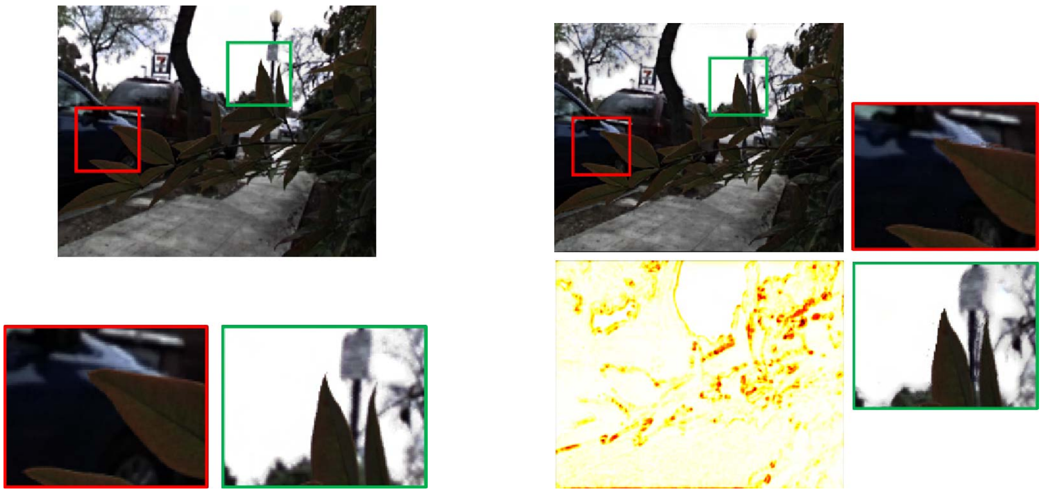
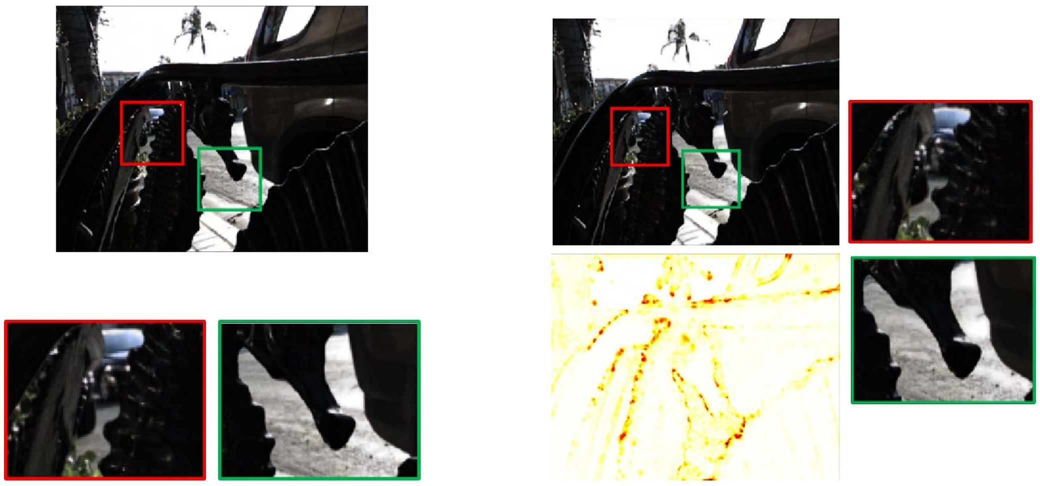

(b) Flower

FIGURE 7.10: *Cont.*



(c) Leaves



(d) Seahorse

Ground Truth Our result

FIGURE 7.10: Visual analysis for the ‘Car’, ’Flower‘, ’Leaves‘ and ’Seahorse‘ images’ synthesized left most horizontal sub-aperture view and the SSIM map with the ground-truth sub-aperture view for the proposed algorithm.

TABLE 7.6: Quantitative comparison for top-left synthesized view comparison for images show in Fig. 7.9.

	Boxes	Cotton	Dino	Sideboard
	Our result			
PSNR	27.52	43.07	35.92	25.83
SSIM	0.9137	0.9925	0.9808	0.9431
	Kalantari *et al.* [10]			
PSNR	19.77	17.40	19.44	20.04
SSIM	0.8417	0.9150	0.9420	0.9324
	Chao *et al.* [11]			
PSNR	23.72	24.63	19.32	19.51
SSIM	0.7938	0.8955	0.8614	0.6102

TABLE 7.7: Comparison with Wu *et al.* [12], Yeung *et al.* [13] and Kalantari *et al.* [10] for the 30 scene dataset

30 Scenes dataset	**Our result**	**Wu *et al.* [12]**	**Yeung *et al.* [13]**	Kalantari *et al.* [10]
PSNR	36.24	41.02	40.93	37.50
SSIM	0.9922	0.9968	0.98.27	0.97

TABLE 7.8: PSNR and SSIM for the four images shown in Fig.7.10 from the 30 scene dataset

30 Scenes dataset	**Car**	**Flower**	**Leaves**	**Seahorse**
PSNR	30.02	30.98	28.27	31.04
SSIM	0.9921	0.9877	0.9786	0.9923

the depth values range from +2 to -2, as opposed to synthetic images, which have depth values ranging from +4 to -4. Since our algorithm is a non-learning-based approach, the only change we make to synthesize real light field images is to change the depth range, which shows the flexibility of our approach. It can be seen from the results shown in Table 7.7 that our approach produces comparable results in terms of the SSIM but reduces in accuracy in terms of the PSNR values. The reduction is because Wu *et al.* [12] takes 3 x 3 input images and only interpolates one image between their input views. Kalantari *et al.* [10] use the corner images as input and interpolate all the internal views, while Yeung *et al.* [13] in their 2 x 2 - 8 x 8 set-up only extrapolate two views in both directions, while the other views are interpolated within the baseline of the input images. On the other hand, we only use the central image as input and extrapolate three images in both directions to synthesize a 7 x 7 light field image. We mainly use the focal stack image and the all-in-focus image as input instead of sub-aperture views because sub-aperture views are comparatively more difficult to capture. In addition, while extrapolating to synthesize the views using sub-aperture images, we don't have any information to fill the occluded regions.

Fig. 7.10 shows the visual comparison for the left-topmost sub-aperture image with the ground truth for four images from the 30-Scenes dataset [10]. Table 7.8 shows the PSNR and SSIM results for the four images shown in Fig. 7.10 from the 30-Scenes dataset [10]. We have chosen these images as there is a significant depth difference between the foreground and background regions in these images. In Fig. 7.10(a), the bark of the tree covers part of the road and the car. It can be seen from the magnified images that the bark in the synthesized image shows no blurring around the edges near the road or the car, as seen in the red and green magnified images, respectively. The flower scene in Fig. 7.10(b) consists of plants and trees in the foreground and cars, houses and a man in the background. The magnified images in Fig. 7.10(b) show that the edges of the leaves are sharp, and even the bark with the house in the background is synthesized correctly. But closer inspection of the image reveals that just to the left of the green magnification window, the edges of the window on the house in the background slant a little to the right. An error of the depth map causes this abnormality in the synthesized view. A similar aberration can be seen in Fig. 7.10(c) in the magnified green window, but this irregularity is due to the incorrect filling of the occluded region of the image. This irregularity can also be seen in the SSIM map, highlighted by the dark red spots. In Fig. 7.10(d), the red magnification window shows no blurring effect near the seahorse's snout, but if we look closely at the green magnification window, we notice that the gap between the seahorse and the chair handle is less than seen in the ground truth image. This is again due to an error with the depth map, as the car in the background or the seahorse is estimated at a slightly incorrect depth, causing the objects to appear closer.

7.3.4 Data availability

The results presented in this Chapter and our code are available at `https://github.com/rishabhsharma27/Light_field_synthesis_results`.

7.4 Discussion and Conclusion

In this chapter we proposed an algorithm that uses the focal stack images and the all-in-focus image to synthesize a 9 x 9 sub-aperture view light field image. We use depth from defocus to estimate a depth map using the approach presented in Chapter 5. This depth map and the all-in-focus image are then used to synthesize the sub-aperture views. We show that our algorithm can synthesize high-accuracy light field images even with a varying number of focal stack images. We also show that the information revealed from the defocus blur in the focal stack image of regions not visible in the all-in-focus image can be used to fill the gaps in occluded regions in both the horizontal and vertical synthesized views. In our approach, using the frequency domain to mimic the apparent movement of the regions at different depths in the sub-aperture view ensures sub-pixel accuracy even for small depth values.

Fig. 7.6 in Section 7.2.3.3 represents the order in which the light field image is synthesized. We start at the blue square in Fig. 7.6, representing the all-in-focus image and the estimated depth map and synthesize the central horizontal and vertical sub-aperture views, which in this depth map is represented by the green and yellow squares. Thus, the orange squares can be synthesized using the central horizontal and vertical sub-aperture views. As we use the same depth map and synthesize the central horizontal and vertical sub-aperture views, the error in the synthesized view will occur only due to an error in the estimated depth map or the occluded regions. But since the views represented by the orange squares are synthesized from both the horizontal and vertical direction, we average them to reduce the errors in the final synthesized light field image.

It can be seen from the visual comparison shown in Fig. 7.7, Fig. 7.9, and comparative quantitative results presented in Table 7.2 to 7.6 that our proposed algorithm outperforms the three algorithms we have studied for both the PSNR and SSIM metrics. One main disadvantage of Kalantari *et al.* [10] is that they use four corner sub-aperture views for synthesis, and it is not easy to capture the corner views without moving the camera. Chao *et al.* [11] uses a large baseline horizontal stereo pair and interpolates the horizontal views within that baseline. Still, as no information is available for the extrapolated vertical views, the algorithm's accuracy reduces for the corner sub-aperture views.

Furthermore, for our algorithm, the final resolution of the light field image mainly depends on the resolution of the central all-in-focus image. The precision of the depth map only ensures parallax accuracy in the sub-aperture views. So even if the depth map precision is reduced, that will only reduce the amount of parallax of the synthesized light field image. Still, the resolution of the light field will correspond to the central view's resolution.

Wu *et al.* [12], and Kalantari *et al.* [10] only interpolate images within the baseline of the input image to synthesize the internal views. Yeung *et al.* [13] in their 2 x 2 - 8 x 8 set-up only extrapolate two views in each direction, while the other views are interpolated within the baseline of input images. Since these algorithms interpolate most of the synthesized views between the baseline of the input views, they achieve higher accuracy than our approach, but as these algorithms require sub-aperture views as input, these algorithms are not practical for light field synthesis using 2D cameras. In contrast, focal stack images can be captured relatively easily as we don't need to use any additional equipment to move the camera to capture different viewpoints instead we only need to change the camera's focal point.

Chapter 8

Future work

For the depth estimation techniques described in Chapter 5, 6, and 7, the number of depth levels in the depth map is proportional to the number of images in the focal stack. We can increase the number of focal stack images for the depth estimation for light field images to improve the depth map precision as the input data is the light field image. However, the same is not possible for depth estimation techniques in Chapter 6 and 7, as the input data for these algorithms is the focal stack and not the light field image. As shown in Fig. 5.17 in Section 5.3, the MSE for the focal stack patches compared to the all-in-focus image patches traces a bell curve, where the out-of-focus patches almost fall on a straight line, whereas, as the patches approach the focused patch we see a bell curve. In our future work, we intend to analyze the patches that fall on the bell curve and check how the MSE for the patches changes if we apply a blurring filter by gradually increasing the order of blurring. This analysis would help us understand at what point the patch blurs enough that the MSE falls on a straight line and accordingly decide the depth of that patch.

For the light field synthesis presented in Chapter 7, the number of depth levels in the depth map is dependent on the number of images in the focal stack, so fewer focal stack images produce abrupt discontinuities for objects with two or more depths, as shown in Fig. 8.1 (highlighted by the red squares). The image shown in Fig. 8.1 is generated using five focal stack images. In our future work, we intend to use the focal stack images to estimate the amount of blur for the same defocus regions between consecutive focal stack images

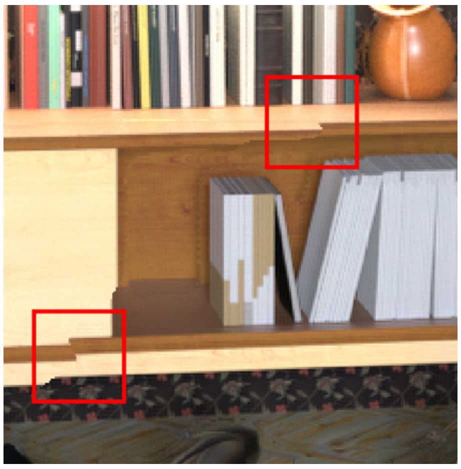


FIGURE 8.1: Error in light field synthesis using fewer images in the focal stack.

and use that information to increase the depth levels of the depth map. Increasing the depth levels using fewer focal stack images will reduce the effect of abrupt discontinuities for objects with two or more depths in the synthesized view, increasing the synthesized view accuracy with fewer focal stack images. In our future work, for light field synthesis using focal stack images captured by a 2D camera, we intend to compare our algorithm accuracy with RVS and VSRS view synthesis algorithms [100] that use DIBR.

Our current work only verifies the light field synthesis approach using the focal stack generated with the light field images. This makes it easy to verify the accuracy of the synthesized views by comparing them to the sub-aperture views of the light field image. In future work, the depth estimation techniques described in Chapter 6 could be used with the focal stack captured by a 2D camera to synthesize the light field image. However, as the light field image would not be available, we would not be able to compare the accuracy of the synthesized views. To solve this, we propose a creative approach to capture the light field image with a plenoptic camera, and then use a 2D camera to capture a focal stack and all-in-focus image of the same scene. Then, align the all-in-focus image with the central view of the light field image. Using the same translation parameters, finally align all of the focal stack images. The synthesized light field sub-aperture views would then be in-line with the plenoptic camera sub-aperture views. Even though the color of the images captured with the two cameras would be different, it is still possible to check structural similarity to estimate the accuracy of the synthesized light field image.

Some commercial cameras such as Lumix [101] and Olympus [93] already have a feature called focus stacking that can take high-resolution focal stack images and merge them to create a sharper all-in-focus image. These cameras also allow the user to save the individual focal stack images in the raw format. Thus, with further development our algorithm can help to enable light field creation. In our future work, we intend to use these focal stack images and all-in-focus images to synthesize the light field image with an angular resolution of 15 x 15 with a high spatial resolution of the individual sub-aperture views. We also intend to capture the focal stack with a 2D camera and align the light field camera with the 2D camera to capture a light field of the same scene and use that as reference views to check the accuracy of our approach.

Chapter 9

Conclusion

Light field images capture rich visual information by representing light distribution in free space, allowing researchers to enhance computer vision applications' performance, such as depth estimation, post-capture refocusing, and image segmentation. Depth maps from light field images play a crucial role in applications like light field image compression techniques, reconstructing views from a sparse set of perspective views, increasing the number of perspective views, and 3D reconstruction.

This thesis focused on novel contributions to two main issues for depth estimation algorithms for light field images: depth inaccuracies around occlusions, and depth map accuracy for noisy images. We also extended our depth estimation algorithm to synthesize the light field image.

To address occlusion and noise-resilient depth map estimation, we proposed a depth-from-defocus algorithm that uses the focal stack and all-in-focus image. To ensure depth accurately around depth discontinuities, we used the median value of the shifted sub-aperture images instead of taking the average value to generate the focal stack. This reduced the defocus blur around the depth discontinuities, improving our algorithm's accuracy near depth boundaries. We showed that using frequency domain analysis instead of the RGB images to compare the focal stack image regions with the all-in-focus image improved our algorithm's resilience to noise.

Then, with a few modifications to our depth estimation algorithm for light field images, we can extend it for depth estimation for a focal stack captured by a 2D camera. We evaluated different parameters for capturing the focal stack, and showed that using an f-stop of f/1.8 and a step size of 100 covered the scene with sufficient depth levels to synthesize a light field image using the depth map and all-in-focus image.

To evaluate our proposed depth map estimation approach in a real-world application, we presented a novel light field synthesis algorithm that improves the synthesis accuracy near depth discontinuities. Our algorithm uses the focal stack images and the all-in-focus image to synthesize a 9 x 9 sub-aperture view light field image. Using the depth-from-defocus approach, we first synthesize the depth map and use this depth map with the all-in-focus image to synthesize the sub-aperture views. We showed that by using the frequency domain to mimic the apparent movement of the regions at different depths in the sub-aperture view, and using the information extracted from the blurred regions of the focal stack to fill the occluded regions, we ensured the high accuracy of the synthesized views.

Finally, to address the challenge that existing light field datasets do not comprehensively contain the light field image, focal stack, ground-truth depth, and disparity map, we proposed and implemented an extension to existing work on a light field camera simulation model in Blender. This extended toolkit enables researchers to evaluate algorithms for both depth map estimation and light field synthesis. In evaluating light field synthesis algorithms, the focal stack of the light field image can be captured with varying camera parameters such as image resolution, f-stop and depth of field. In addition to enabling the evaluations of our proposed depth map and light field synthesis approaches presented in this thesis, our extension to the toolkit will also enable advancements in CNN and deep learning approaches that use focal stack images for light field synthesis.

Bibliography

[1] Marc Levoy and Pat Hanrahan. Light field rendering. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 441–452. 2023.

[2] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 765–776. ACM, 2005. ISBN 0730-0301.

[3] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III 13*, pages 19–34. Springer, 2017.

[4] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11):1–11, 2005.

[5] Michael Strecke, Anna Alperovich, and Bastian Goldluecke. Accurate depth and normal maps from occlusion-aware focal stack symmetry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2814–2822, 2017.

[6] T. Wang, A. A. Efros, and R. Ramamoorthi. Occlusion-aware depth estimation using light-field cameras. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3487–3495. ISBN 2380-7504. doi: 10.1109/ICCV.2015.398.

[7] S. Zhang, H. Sheng, C. Li, J. Zhang, and Z. Xiong. Robust depth estimation for light field via spinning parallelogram operator. *Computer Vision and Image Understanding*, 145:148–159, 2016. doi: 10.1016/j.cviu.2015.12.007.

[8] Changha Shin, Hae-Gon Jeon, Youngjin Yoon, In So Kweon, and Seon Joo Kim. Epinet: A fully-convolutional neural network using epipolar geometry for depth from light field images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4748–4757, 2018.

[9] Zhoutong Zhang, Yebin Liu, and Qionghai Dai. Light field from micro-baseline image pair. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3800–3809, 2015.

[10] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[11] Chun-Hao Chao, Chang-Le Liu, and Homer H Chen. Robust light field synthesis from stereo images with left-right geometric consistency. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1844–1848. IEEE, 2021.

[12] Gaochang Wu, Yebin Liu, Lu Fang, Qionghai Dai, and Tianyou Chai. Light field reconstruction using convolutional network on epi and extended applications. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1681–1694, 2018.

[13] Henry Wing Fung Yeung, Junhui Hou, Jie Chen, Yuk Ying Chung, and Xiaoming Chen. Fast light field reconstruction with deep coarse-to-fine modeling of spatial-angular clues. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 137–152, 2018.

[14] Rishabh Sharma, Stuart Perry, and Eva Cheng. Noise-resilient depth estimation for light field images using focal stack and fft analysis. *Sensors*, 22(5):1993, 2022.

[15] Rishabh Sharma, Stuart Perry, and Eva Cheng. Light field view synthesis using the focal stack and all-in-focus image. *Sensors*, 23(4):2119, 2023.

[16] Gaochang Wu, Belen Masia, Adrian Jarabo, Yuchen Zhang, Liangyong Wang, Qionghai Dai, Tianyou Chai, and Yebin Liu. Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):926–954, 2017. ISSN 1932-4553.

[17] Antoine Mousnier, Elif Vural, and Christine Guillemot. Partial light field tomographic reconstruction from a fixed-camera focal stack. *arXiv preprint arXiv:1503.01903*, 2015.

[18] Yasutaka Inagaki, Yuto Kobayashi, Keita Takahashi, Toshiaki Fujii, and Hajime Nagahara. Learning to capture light fields through a coded aperture camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.

[19] Jianwei Zhou, Da Yang, Zhenglong Cui, Sizhe Wang, and Hao Sheng. Lrfnet: An occlusion robust fusion network for semantic segmentation with light field. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1178–1168. IEEE, 2021.

[20] Xinxin Hu, Kailun Yang, Lei Fei, and Kaiwei Wang. Acnet: Attention based network to exploit complementary features for rgbd semantic segmentation. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1440–1444. IEEE, 2019.

[21] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002. ISSN 0920-5691.

[22] Can Chen, Haiting Lin, Zhan Yu, Sing Bing Kang, and Jingyi Yu. Light field stereo matching using bilateral statistics of surface cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1518–1525, 2014.

[23] Y. Zhang, H. Lv, Y. Liu, H. Wang, X. Wang, Q. Huang, X. Xiang, and Q. Dai. Light-field depth estimation via epipolar plane image analysis and locally linear embedding. *IEEE Transactions on Circuits and Systems for Video Technology*, 27 (4):739–747, 2017. ISSN 1051-8215. doi: 10.1109/TCSVT.2016.2555778.

[24] Yoav Y. Schechner and Nahum Kiryati. Depth from defocus vs. stereo: How different really are they? *Int. J. Comput. Vision*, 39(2):141–162, 2000. ISSN 0920-5691. doi: 10.1023/a:1008175127327.

[25] M. Feng, Y. Wang, J. Liu, L. Zhang, H. F. M. Zaki, and A. Mian. Benchmark data set and method for depth estimation from light field images. *IEEE Transactions on Image Processing*, 27(7):3586–3598, 2018. ISSN 1057-7149. doi: 10.1109/TIP.2018.2814217.

[26] S. Heber, W. Yu, and T. Pock. Neural epi-volume networks for shape from light field. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2271–2279. ISBN 2380-7504. doi: 10.1109/ICCV.2017.247.

[27] Lixin Shi, Haitham Hassanieh, Abe Davis, Dina Katabi, and Fredo Durand. Light field reconstruction using sparsity in the continuous fourier domain. *ACM Transactions on Graphics (TOG)*, 34(1):1–13, 2014.

[28] Anat Levin and Fredo Durand. Linear view synthesis using a dimensionality gap light field prior. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1831–1838. IEEE, 2010.

[29] Gaochang Wu, Belen Masia, Adrian Jarabo, Yuchen Zhang, Liangyong Wang, Qionghai Dai, Tianyou Chai, and Yebin Liu. Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):926–954, 2017.

[30] Sven Wanner and Bastian Goldluecke. Variational light field analysis for disparity estimation and super-resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):606–619, 2013.

[31] Akira Kubota, Kiyoharu Aizawa, and Tsuhan Chen. Reconstructing dense light field from array of multifocus images for novel view synthesis. *IEEE Transactions on Image Processing*, 16(1):269–279, 2006.

[32] Antoine Mousnier, Elif Vural, and Christine Guillemot. Partial light field tomographic reconstruction from a fixed-camera focal stack. *arXiv preprint arXiv:1503.01903*, 2015.

[33] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2243–2251, 2017.

[34] Ole Johannsen, Katrin Honauer, Bastian Goldluecke, Anna Alperovich, Federica Battisti, Yunsu Bok, Michele Brizzi, Marco Carli, Gyeongmin Choe, Maximilian Diebold, et al. A taxonomy and evaluation of dense light field depth estimation algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 82–99, 2017.

[35] Ren Ng. *Digital light field photography.* stanford university California, 2006. ISBN 0542707799.

[36] Edward H Adelson and James R Bergen. The plenoptic function and the elements of early vision. 1991.

[37] Edward H Adelson and John YA Wang. Single lens stereo with a plenoptic camera. *IEEE transactions on pattern analysis and machine intelligence*, 14(2):99–106, 1992.

[38] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23 (11):1222–1239, 2001.

[39] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26 (2):147–159, 2004.

[40] 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*, pages 82–96. Springer, 2002.

[41] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2115–2128, 2009. ISSN 0162-8828.

[42] Michael Bleyer, Carsten Rother, and Pushmeet Kohli. Surface stereo with soft segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1570–1577. IEEE. ISBN 1424469856.

[43] Yichen Wei and Long Quan. Asymmetrical occlusion handling using graph cut for multi-view stereo. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 902–909. IEEE, 2005.

[44] Oliver J Woodford, Ian D Reid, Philip HS Torr, and Andrew W Fitzgibbon. On new view synthesis using multiview stereo. In *BMVC*, volume 2, pages 1120–1129, 2007.

[45] Abhijit S Ogale and Yiannis Aloimonos. Stereo correspondence with slanted surfaces: critical implications of horizontal slant. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[46] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 399–406. IEEE, 2005.

[47] Zhan Yu, Xinqing Guo, Haibing Lin, Andrew Lumsdaine, and Jingyi Yu. Line assisted light field triangulation and stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2792–2799, 2013.

[48] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary mrfs via extended roof duality. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[49] Jingyi Yu, Leonard McMillan, and Steven Gortler. Surface camera (scam) light field rendering. *International Journal of Image and Graphics*, 4(04):605–625, 2004.

[50] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998.

[51] Engin Tola, Christoph Strecha, and Pascal Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5): 903–920, 2012. ISSN 0932-8092.

[52] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2009.

[53] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[54] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[55] Robert C Bolles, H Harlyn Baker, and David H Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision*, 1(1):7–55, 1987. ISSN 0920-5691.

[56] O. Johannsen, A. Sulc, and B. Goldluecke. What sparse light field coding reveals about scene structure. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3262–3270. ISBN 1063-6919. doi: 10.1109/CVPR.2016.355.

[57] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Transactions on Graphics*, 32(4), 2013. doi: 10.1145/2461912.2461926.

[58] Sven Wanner and Bastian Goldluecke. *Globally Consistent Depth Labeling of 4D Light Fields*. 2012. ISBN 978-1-4673-1226-4. doi: 10.1109/CVPR.2012.6247656.

[59] S. Wanner and B. Goldluecke. Reconstructing reflective and transparent surfaces from epipolar plane images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8142 LNCS, pages 1–10. 2013. doi: 10.1007/978-3-642-40602-7_1.

[60] Antonio Criminisi, Sing Bing Kang, Rahul Swaminathan, Richard Szeliski, and P. Anandan. Extracting layers and analyzing their specular properties using epipolar-plane-image analysis. *Comput. Vis. Image Underst.*, 97(1):51–85, 2005. ISSN 1077-3142. doi: 10.1016/j.cviu.2004.06.001.

[61] X. Zhu, S. Cohen, S. Schiller, and P. Milanfar. Estimating spatially varying defocus blur from a single image. *IEEE Transactions on Image Processing*, 22(12):4879–4891, 2013. ISSN 1057-7149. doi: 10.1109/TIP.2013.2279316.

[62] Shaojie Zhuo and Terence Sim. Defocus map estimation from a single image. *Pattern Recognition*, 44(9):1852–1858, 2011. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2011.03.009.

[63] S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994. ISSN 0162-8828. doi: 10.1109/34.308479.

[64] P. Favaro, S. Soatto, M. Burger, and S. J. Osher. Shape from defocus via diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):518–531, 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1175.

[65] H. Lin, C. Chen, S. B. Kang, and J. Yu. Depth recovery from light field using focal stack symmetry. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3451–3459. ISBN 2380-7504. doi: 10.1109/ICCV.2015.394.

[66] R. R. Sahay and A. N. Rajagopalan. Harnessing defocus blur to recover high-resolution information in shape-from-focus technique. *IET Computer Vision*, 2(2): 50–59, 2008. ISSN 1751-9632. doi: 10.1049/iet-cvi:20070072.

[67] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *2013 IEEE International Conference on Computer Vision*, pages 673–680. ISBN 1550-5499. doi: 10.1109/ICCV.2013.89.

[68] S. Heber and T. Pock. Convolutional networks for shape from light field. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3746–3754. ISBN 1063-6919. doi: 10.1109/CVPR.2016.407.

[69] 3dmd laser scanner. *Mar. 2018.*

[70] Lei Han, Xiaohua Huang, Zhan Shi, and Shengnan Zheng. Depth estimation from light field geometry using convolutional neural networks. *Sensors*, 21(18):6061, 2021.

[71] Stefan Heber and Thomas Pock. Convolutional networks for shape from light field. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3746–3754, 2016.

[72] Chunle Guo, Jing Jin, Junhui Hou, and Jie Chen. Accurate light field depth estimation via an occlusion-aware network. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[73] Jinglei Shi, Xiaoran Jiang, and Christine Guillemot. A framework for learning depth from a flexible subset of dense and sparse light field views. *IEEE Transactions on Image Processing*, 28(12):5867–5880, 2019.

[74] Titus Leistner, Hendrik Schilling, Radek Mackowiak, Stefan Gumhold, and Carsten Rother. Learning to think outside the box: Wide-baseline light field depth estimation with epi-shift. In *2019 International Conference on 3D Vision (3DV)*, pages 249–257. IEEE, 2019.

[75] Jiaxin Chen, Shuo Zhang, and Youfang Lin. Attention-based multi-level fusion network for light field depth estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1009–1017, 2021.

[76] Yu-Ju Tsai, Yu-Lun Liu, Ming Ouhyoung, and Yung-Yu Chuang. Attention-based view selection networks for light-field disparity estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12095–12103, 2020.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[79] Jiayong Peng, Zhiwei Xiong, Yicheng Wang, Yueyi Zhang, and Dong Liu. Zero-shot depth estimation from light field using a convolutional neural network. *IEEE Transactions on Computational Imaging*, 6:682–696, 2020.

[80] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[81] Anna Alperovich, Ole Johannsen, Michael Strecke, and Bastian Goldluecke. Light field intrinsics with a deep encoder-decoder network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9145–9154, 2018.

[82] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, 1993.

[83] Chang Liu, Jun Qiu, and Ming Jiang. Light field reconstruction from projection modeling of focal stack. *Optics express*, 25(10):11377–11388, 2017.

[84] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2243–2251, 2017.

[85] Anil Kumar Vadathya, Sharath Girish, and Kaushik Mitra. A unified learning-based framework for light field reconstruction from coded projections. *IEEE Transactions on Computational Imaging*, 6:304–316, 2019.

[86] Fernando Pérez, Alejandro Pérez, Manuel Rodríguez, and Eduardo Magdaleno. Lightfield recovery from its focal stack. *Journal of Mathematical Imaging and Vision*, 56:573–590, 2016.

[87] M Shahzeb Khan Gul, M Umair Mukati, Michel Bätz, Søren Forchhammer, and Joachim Keinert. Light-field view synthesis using a convolutional block attention module. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3398–3402. IEEE, 2021.

[88] Huachun Wang, Binbin Yan, Xinzhu Sang, Duo Chen, Peng Wang, Shuai Qi, Xiaoqian Ye, and Xiao Guo. Dense view synthesis for three-dimensional light-field displays based on position-guiding convolutional neural network. *Optics and Lasers in Engineering*, 153:106992, 2022.

[89] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[90] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[91] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[92] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[93] Olympus camera focus stacking. *accessed on 12 November 2022.*

[94] Martin Rerabek and Touradj Ebrahimi. New light field image dataset. In *8th International Conference on Quality of Multimedia Experience (QoMEX)*, number CONF, 2016.

[95] Kshitij Marwah, Gordon Wetzstein, Yosuke Bando, and Ramesh Raskar. Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[96] Sven Wanner, Stephan Meister, and Bastian Goldluecke. Datasets and benchmarks for densely sampled 4d light fields. In *VMV*, volume 13, pages 225–226. Citeseer, 2013.

[97] Donald G Dansereau, Oscar Pizarro, and Stefan B Williams. Decoding, calibration and rectification for lenselet-based plenoptic cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1027–1034, 2013.

[98] David E Jacobs, Jongmin Baek, and Marc Levoy. Focal stack compositing for depth of field control. *Stanford Computer Graphics Laboratory Technical Report*, 1(1):2012, 2012.

[99] Sharpness estimation from image gradients,matlab central file exchange. `https://www.mathworks.com/matlabcentral/fileexchange/32397-sharpness-estimation-from-image-gradients`. Accessed: 2021-1-28.

[100] Sarah Fachada, Daniele Bonatto, Arnaud Schenkel, and Gauthier Lafruit. Depth image based view synthesis with multiple reference views for virtual reality. In *2018-3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4. IEEE, 2018.

[101] Panasonic camera focus stacking. *accessed on 12 November 2022*.