Real-time Depth Estimation from Light Fields on Embedded Hardware

Thesis approved by
the Department of Computer Science
University of Kaiserslautern-Landau
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)
to
Yuriy Anisimov

Date of Defense: 15.03.2024
Dean: Prof. Dr. Christoph Garth
Reviewer: Prof. Dr. Didier Stricker
Reviewer: Prof. Dr. Karsten Berns

DE-386

Abstract

In recent years, there has been a growing need for accurate three-dimensional (3D) scene reconstruction. Recent developments in the automotive industry have led to the increased use of advanced driver-assistance system (ADAS) where 3D reconstruction techniques are used, for example, as part of a collision detection system. For such applications, scene geometry reconstruction is usually performed in the form of depth estimation, where distances to scene objects are obtained.

In general, depth estimation systems can be divided into active and passive. Both systems have their advantages and disadvantages, but passive systems are usually cheaper to produce and easier to assemble and integrate than active systems. Passive systems can be stereo- or multiple-view based. Up to a certain limit, increasing the number of views in multi-view systems usually results in improved depth estimation accuracy.

One potential problem for ensuring the reliability of multi-view systems is the need to accurately estimate the orientation of their optical sensors. One way to ensure sensor placement for multi-view systems is to rigidly fix the sensors at the manufacturing stage. Unlike arbitrary sensor placement, using of a simplified and known sensor placement geometry further simplifies the depth estimation.

We meet with the concept of light field (LF), which parameterizes all visible light passing through all viewpoints by their intersection with angular and spatial planes. When applied to computer vision, this gives us a two-dimensional (2D) set of 2D images, where the physical distances between each image are fixed and proportional to each other.

Existing LF depth estimation methods provide good accuracy, which is suitable for industrial applications. However, the main problems of these methods are related to their running time and resource requirements. Most of the algorithms presented in the literature are typically sharpened for accuracy, can only be run on high-performance machines and often require a significant amount of time to process and obtain results.

Real-world applications often have running time requirements. Also, often there is a power-consumption limitation. In this dissertation, we investigate the problem of building a depth estimation system with an LF camera that satisfies the operating time and power consumption constraints without significant loss of estimation accuracy.

First, an algorithm for calibrating LF cameras is proposed, together with an algorithm for automatic calibration refinement, that works on arbitrary captured scenes. An algorithm for classical geometric depth estimation using LF cameras is proposed. Ways to optimize the algorithm for real-time use without significant loss of accuracy are presented. Finally, the ways how the presented depth estimation methods can be extended using modern deep learning paradigms under the two previously mentioned constraints are shown.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Dr. Didier Stricker, for his invaluable guidance and support throughout my dissertation. His expertise and insights have been instrumental in shaping my research and helping me to develop as a researcher. Also, I would like to thank Prof. Dr. Berns for reviewing my dissertation and Prof. Dr. van Waveren for being a chairman of my viva.

I would also like to thank Dr. Oliver Wasenmüller for his collaboration on the project that formed the core of my dissertation. His dedication, expertise, and enthusiasm for the project were essential to its success, and I am grateful for the opportunity to have worked with him.

I would like to acknowledge the Department of Computer Science of RPTU and German Research Center for Artificial Intelligence for providing me with a stimulating and supportive academic environment. The resources, facilities, and staff of the department have been instrumental in enabling me to conduct my research and to develop as a researcher.

I would also like to express my appreciation to my colleagues and students, who provided me with invaluable feedback, support, and inspiration throughout my academic journey. Their encouragement and enthusiasm have been essential to my success, and I am grateful for the opportunity to work with such a talented and dedicated group of individuals.

I would like to express special thanks to David Haase, whose help in setting up the hardware has bailed me out more than once, and with whom conversations have become an outlet from routine work.

I would like to thank Nikolai Markov, who was my supervisor during my studies in Moscow, who introduced me to computer vision, and thanks to whom I decided to continue my studies in this direction.

Finally, I would like to thank my friends Aleksandr, Aleksei, Kirill and Sergei for their friendship and brotherhood, and my mother Tatiana for everything good she has done. I must express my deepest gratitude to my wife Polina, whose support and faith helped me to finish this dissertation.

Thank you all for your contributions to my research and for helping me to achieve my academic goals.

Yuriy Anisimov, 31.03.2024

Contents

List of Figures

List of Tables

List of Abbreviations

1D one-dimensional 12

2D two-dimensional iii, xi, 8–13, 16, 19, 20, 25, 34, 42, 43, 61, 62, 73

3D three-dimensional iii, xi, 1–3, 8–12, 14, 16–19, 25, 27–31, 33, 34, 37, 41–44, 47, 48, 65, 89, 109, 113, 118

4D four-dimensional 11, 19, 66

4DLFB 4D light field benchmark 87

5D five-dimensional 18, 19

AC after Christ 5, 6

ADAS advanced driver-assistance system iii, xiii, 1, 105, 107, 108

API application programming interface 22, 91

BC before Christ 5

CCD charge-coupled device 6, 7

CCS camera coordinate system 10, 11

CMOS complementary metal-oxide-semiconductor 6, 7

CNN convolutional neural network 34, 37

CPU central processing unit 84, 86, 99, 101

CUDA compute unified device architecture 86, 98, 99

DLT direct linear transformation 43, 44

DoF degrees of freedom 28

DoG difference of Gaussians 49

EPI epipolar plane image 28–34, 38, 39

FPS frames per second 25

GFTT good features to track 46, 49

GPGPU general purpose computing on graphics processing unit 98

GPU graphics processing unit 2, 36, 86, 88, 98, 99, 101, 102

GS gold standard 50, 59

ICP iterative closest points 76

KDE kernel density estimation 30

KLT Kanade–Lucas–Tomasi feature tracker 36, 46, 49

LF light field iii, xi–xiii, xv, 2, 3, 5, 12, 13, 17, 19–47, 49, 54–57, 59–61, 66–68, 70, 71, 76–79, 81–85, 87, 89, 91, 94, 96, 98, 101, 106, 108, 109, 112, 113, 116–122

LLS linear least squares 30, 32, 43

LMA Levenberg-Marquardt algorithm 43, 44, 53, 54

LUT lookup table 13, 45, 97, 101

MLA micro-lens array 24

MOS metal–oxide–semiconductor 7

MRF Markov random field 30, 35, 36

MSE mean squared error xv, 79, 82

NLLS nonlinear least squares 31

PC personal computer 29

PnP perspective-n-point 44

PSO particle swarm optimization 53

RAM random access memory 99, 109

RANSAC random sample consensus 47, 50

RGB red-green-blue xi, 1, 19, 22, 23, 96, 105–107, 117

SAD sum of absolute differences 32

SGM semi-global matching xii, 72–74, 79, 80, 84, 101

SGrD sum of gradient differences 32

SIFT Scale-Invariant Feature Transform 49, 59

SoA state-of-the-art 27, 40

SSD sum of squared differences 35

SVD singular value decomposition 48, 52, 53, 60

SVM support vector machine 33

WCS world coordinate system 8, 10–12

WTA winner-takes-all 67

ZNCC zero-mean normalized cross-correlation 32, 36

Introduction

The objective of this work is to study, how to design a system for accurate 3D reconstruction that can be used in environments with special requirements for fast computation times and reduced power consumption. Most works related to this topic either focus more often on the aspect of maximum accuracy, or in few cases only on the execution time of the computations, so that research into the combination of the two requirements is relatively rare.

1.1 Motivation

In the recent years, the demand for accurate and fast 3D reconstruction systems has been at its peak and continues to grow. One of the most notable possible applications for systems with fast reconstruction times are various systems within the group of ADAS. In such systems, fast reconstruction times, together with high accuracy, are essential. For example, in a collision avoidance system, we would want to know the exact distance to a possible object as early as possible for the sake of collision avoidance.

In general, 3D reconstruction systems can be divided into active and passive. In active systems, the device that captures the scene provides data directly about the geometry of the scene, while in passive systems this data is represented indirectly and must be retrieved. From this point of view, active systems would be the preferred tool to use, but the relative cost and complexity of implementation compared to passive systems might incline us to choose the latter.

Passive 3D reconstruction systems usually consist of two or more perceptual devices, which are generally considered to be RGB cameras. Systems with two sensors are usually called "stereo-view cameras", or shortly "stereo cameras", while systems with more than two sensors are "multiple-view cameras" or "multi-view cameras".

Increasing the number of sensors generally to a certain limit leads to an increase in the accuracy of such systems. But this, in turn, also leads to a more complex system. In particular, for passive 3D reconstruction algorithms, accurate information about the relative position of the sensors in relation to each other is needed. During the system setup and adjustment phase, camera positions can be calculated with a relatively high accuracy, sufficient for reconstruction algorithms. But the preservation of the position of the cameras

during operation cannot be guaranteed due to *e.g.* mechanical influences acting on the system. One way to prevent it is to fix the sensors rigidly so that their displacement relative to each other can either be ignored in operation or can be relatively easily compensated.

The placement of the sensors also plays an important role. In general, with known camera positions we can set some limits for the search areas of the same object (or the same or the same point of the 3D scene) between images from different sensors. Simplifications of this type can have a positive effect on the execution time of the reconstruction algorithm. The simplest in this case seems to be the arrangement of the sensors in a so-called grid, where the position of each sensor is set and can be derived relative to each other, provided that the distance between the sensors is fixed and known

Therefore, ideally, we would like to have a device with a fixed placement of sensors, the configuration of which implies a simplification in the computational procedures for reconstruction. The concept of a LF camera fits under this definition. Each sensor in such a camera is located proportionally, and therefore the theory of LF capturing can be applicable to such a camera.

The problem of energy consumption is critical for various industrial systems, because in the case of stand-alone systems the energy capacity of batteries is limited, and in the case of systems with the possibility of constant energy supply the energy costs should be reduced, and ideally each subsystem of large systems should be optimized in terms of energy consumption. Modern popular hardware for calculations related to reconstruction, made, for example, on the basis of graphics processing unit (GPU), require power in the range of hundreds of watts, which for some systems may be unacceptable, and for others will lead to additional costs for electricity. In this case we usually refer to so-called embedded systems, *i.e.* systems that operate while being embedded in other systems. Certain stringent requirements apply to such systems in terms of their design and energy consumption. Such systems are subject to certain stringent requirements in terms of their design and power consumption, which makes them an ideal choice for reconstruction algorithms, taking the above-mentioned requirements into account.

This dissertation will describe how a fast 3D reconstruction system with low power consumption can be built based on a LF camera, what problems can be encountered when designing it, and how these problems can be solved.

1.2 Contribution

Our main contribution is the development of an end-to-end system for 3D reconstruction using a LF camera that operates in real time and is executed on embedded hardware with relatively low power consumption. In particular, the technical contributions are:

- Development of LF calibration algorithm together with methods of calibration refinement during exploitation and investigation of LF cameras auto-calibration.

- Development of real-time algorithm for 3D reconstruction from LF images, which shows its feasibility on both synthetic and real-world data.
- Investigation of methods for optimization of existing deep learning algorithms for execution on real-world scenes, taking into account hardware and runtime constraints.

1.3 Overview

The dissertation is organized as follows:

- Chapter 2 presents the basic information necessary to understand the subject of the thesis. In particular, the process of image acquisition by a camera, the problem of camera calibration, possible methods of 3D reconstruction, the commonality of the LF concept with multiple-view cameras are covered.
- Chapter 3 presents a survey of existing calibration methods and algorithms for 3D reconstruction from LFs.
- Chapter 4 describes our research related to the LF camera calibration algorithm, the possibilities for updating calibration data under conditions where re-calibration is not possible, and our findings in the area of auto-calibration for such cameras.
- Chapter 5 explains the way in which the algorithm we developed for 3D reconstruction from LF works.
- Chapter 6 describes how the depth reconstruction system has been brought to life and what are its potential applications.
- Chapter 7 shows how the existing deep learning algorithms can be optimized for their usage on real world scenes with real-time constrains.
- Chapter 8 concludes the presented research and shows the potential future work.

1.4 Publications

Most of the research presented in this dissertation has been approved and presented at peer-reviewed scientific conferences or in journals. The following are publications that have emerged from this research:

1. Yuriy Anisimov, and Didier Stricker (2017). Fast and Efficient Depth Map Estimation from Light Fields. In *2017 International Conference on 3D Vision (3DV 2017)*, Qingdao, China, October 10-12, 2017 (pp. 337–346). IEEE Computer Society.

2. Yuriy Anisimov, Oliver Wasenmüller, and Didier Stricker (2019). A Compact Light Field Camera for Real-Time Depth Estimation. In *Computer Analysis of Images and Patterns - 18th International Conference (CAIP 2019)*, Salerno, Italy, September 3-5, 2019, Proceedings, Part I (pp. 52–63). Springer.

3. Yuriy Anisimov, Oliver Wasenmüller, and Didier Stricker (2019). Rapid Light Field Depth Estimation with Semi-Global Matching. In *15th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP 2019)*, Cluj-Napoca, Romania, September 5-7, 2019 (pp. 109–116). IEEE.

4. Yuriy Anisimov, Gerd Reis, and Didier Stricker (2021). Calibration and Auto-Refinement for Light Field Cameras. *Computer Science Research Notes - CSRN*, CSRN 3101, 254-262. Vaclav Scala Union Agency.

5. Yuriy Anisimov, Jason Raphael Rambach, and Didier Stricker (2022). Nonlinear Optimization of Light Field Point Cloud. *MDPI Sensors*, 22(3), 814. MDPI.

Background

This chapter will explain the fundamental ideas that must be comprehended before moving onto the subsequent chapter of this dissertation. In the corresponding sections we will talk about basic concepts, such as a single camera and the formation of a stereo camera, and then the representation of the LF camera as derived from stereo; what is the depth measurement of the scene and what methods exist for this; what restrictions are imposed by the use of the embedded hardware.

2.1 Single Camera

A derivation of a LF capturing system begins by introducing the concept of a single camera with further expansion by adding cameras on different physical axes.

2.1.1 Definition

Nowadays, a "camera" refers to a device consisting of a housing with a light opening on one side and a surface for recording on the other. This opening is called "an aperture". Aperture is generally equipped with a lens to focus the light rays on the recording surface, but early prototype cameras did not necessarily contain it.

2.1.2 Historical background

People have been attempting to depict their environments in a variety of ways since ancient times, using different ways from parietal art to oil paintings. Early attempts to capture light rays can be traced back to the 5th century before Christ (BC), when the Chinese philosopher Mozi, as noted in [107], was describing the inversion of light after passing through a point, which will later be known as a pinhole [175]. In 11th century after Christ (AC) Ibn al-Haytham was doing experiments with pinholes and candles, exploring the paths of light [140]. In 16th century AC Johannes Kepler [32] created a room-size experimental setup with the lens in pinhole, where the image was composed from passed light on the wall. The term *camera obscura* originates from his work. It took a lot of research efforts before the actual scene could be captured with such principles. The earliest known photograph with

camera obscura, known as "View from the Window at Le Gras" was done by Nicéphore Niépce in 19th century AC [51]. This photograph is shown on Fig. 2.1.

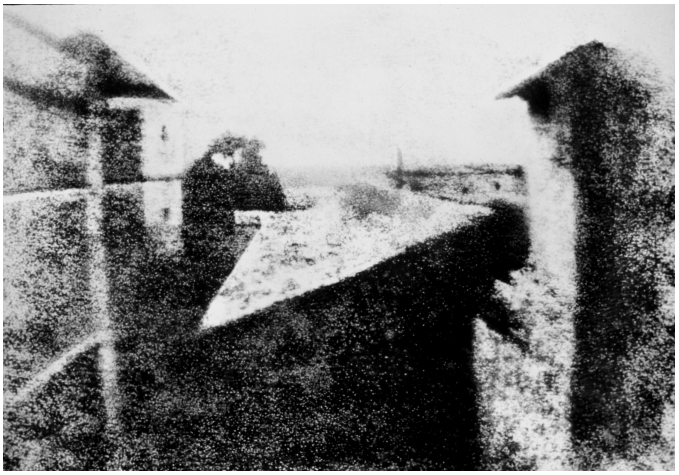


FIGURE 2.1: Enhanced version of "View from the Window at Le Gras". Source: Rebecca A. Moss, Coordinator of Visual Resources and Digital Content Library, College of Liberal Arts Office of Information Technology, University of Minnesota. Public Domain

2.1.3 Components of the modern camera

The devices for image capturing have changed a lot since then. Traditional cameras were using photographic film, made by using emulsion which is sensitive to light, for storing the captured image. To obtain a visible image the film had to be processed by sequentially applying a photographic developer, stop bath solution and fixing solution to it [132].

The intermediate between traditional film cameras and modern cameras are instant cameras (*e.g.* Polaroid). In such cameras, film development takes place directly when the image is captured on the film, and all the chemicals needed to develop the film are applied automatically inside the camera itself. The output of this camera is an already printed photo. Fig. 2.2 shows how the cameras look like. Nowadays, such cameras are hardly ever used for conventional applications. They have been almost completely replaced by digital cameras.

The core component of every digital camera is an image sensor. These sensors consist of the grid of pixels, which perceives a specific color. Usually is it red, green and blue pixels; following the human trichromatic color perception model from Young–Helmholtz theory [176, 22].

There are two main types of digital image sensors: Charge-coupled device (CCD) and Complementary metal-oxide-semiconductor (CMOS) [159]. CCDs were the first developed type of digital sensors. They are based on

(a)

(b)

FIGURE 2.2: An example of (a) film camera and (b) instant camera. Public Domain

metal–oxide–semiconductor (MOS) technology, and every pixel of this sensor is a capacitor. It stores the charge proportionally to the amount of perceived light. Charge of all pixels is converted to the pixel intensity.

In contrast, CMOS sensors consist of the photodiodes with MOS transistors. They serve for the purpose of transferring the charge from one pixel to another, which makes it easier to transmit and convert the electric signal to its digital representation.

Nowadays, CMOS sensors took over the place of CCD. It can be explained by the higher power consumption and lower speed of readout of CCDs compared to CMOS. Another advantage of CMOS sensors is the possibility of integration of the image sensor with other camera components, while CCDs are generally designed as separated components.

Usage of image sensors, which can convert the captured image directly to the digital image is a core of the modern computer vision. Images, captured by the cameras with these sensors, can be streamed or saved in a digital form, allowing online or offline processing of them.

Another important part of a modern camera is a camera lens. They are needed for focusing the incoming light on the camera sensor. Standard type of lenses tries to mimic a human eye perspective by being constructed with

such the focal length, which provides depth of view similar to the human vision [103].

Often, lenses are equipped with a diaphragm. Opening and closing it regularizes the aperture of the lens. By that, the amount of incoming light can be controlled.

2.1.4 Pinhole Model

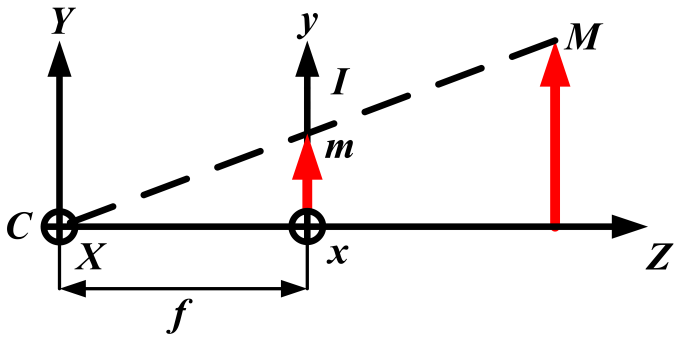


FIGURE 2.3: Simplified representation of pinhole camera model

The principles of *camera obscura* image formation became the basis for what is now known in the literature as "pinhole model" [46]. Its simplified graphical representation is presented on Fig. 2.3. This model describes the core idea of 3D point projection to 2D image coordinate system. Having a point $M = (X, Y, Z)$ in world coordinate system (WCS) the projection $m(\bar{x}, \bar{y})$ of this point to image space I is found as:

$$\bar{x} = f\frac{X}{Z}, \bar{y} = f\frac{Y}{Z}, \tag{2.1}$$

where f stands for a focal length, which is defined as the distance between the center of the lens and the image sensor.

The biggest disadvantage of pinhole camera model derives from the physical properties of our world: in most instances, a camera without a lens cannot be used to focus an image. Thus, the presence of a lens must be taken into account in the problem of projection of light rays.

2.1.5 Lens distortion

Lenses, in turn, introduce various distortions to images. The literature [63] describe two types of such distortions:

- radial distortion, related to lens form and divided into two concepts:
  - barrel distortion, in which the magnification of the image increases with distance from the optical center,
  - pincushion distortion, in which the magnification of the image decreases (opposite to barrel distortion) with distance from the optical center;

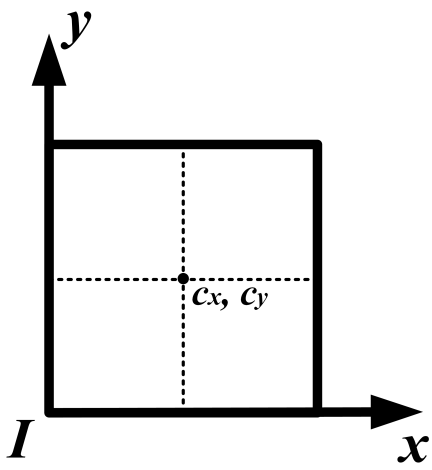


FIGURE 2.4: An image plane with defined camera center

- tangential distortion, related to lens misplacement, when the lens is slightly skewed from being parallel to camera sensor.

While being presented in the literature, effect of the tangential distortion in modern cameras is considered to be negligible and often ignored. Fig. 2.5 demonstrates how the listed distortions look like.

The presence of any of these lens distortions makes it impossible to unambiguously match a 3D point in the scene with its 2D projection. This raises the question: is there any way to compensate for lens distortion to switch to a more comfortable pinhole camera model?

2.1.6 Image projection

Before answering the question from previous chapter, it is needed to discuss the general projection pipeline.

In order to bring the pinhole camera model to use in real camera systems, it needs to be extended based on real camera properties. Here comes the concept of a *perspective camera*: a camera, behavior of which can be modeled by perspective transformation [63].

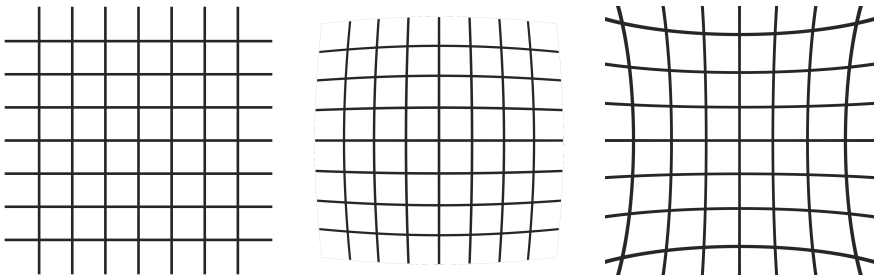
Fig. 2.4 demonstrates the image plane I obtained from the perspective camera, with defined camera center (c_x, c_y). This point creates an offset for the projected points as:

$$y = \bar{y} + c_y, \; x = \bar{x} + c_x \tag{2.2}$$

In order to convert the above coordinates to pixel units, a density of pixels per length unit needs to be taken into consideration.

Since form of a pixel can be non-squared, two separated parameters for every image plane axis are defined as m_x and m_y. Based on that, the actual focal length for every axis is defined as:

$$f_u = f m_x, \; f_v = f m_y \tag{2.3}$$



(a) (b) (c)

FIGURE 2.5: A representation of different lens distortion types: (a) grid without distortion, (b) grid with applied barrel distortion, (c) grid with applied pincushion distortion

In a similar manner, camera center is transformed as:

$$c_u = c_x m_x, \; c_v = c_y m_y \tag{2.4}$$

This conversions allow us to compute the actual pixel coordinate of a projected point (v_d, u_d).

$$u_d = f_u \frac{X}{Z} + c_u, \; v_d = f_v \frac{Y}{Z} + c_v \tag{2.5}$$

Focal lengths f_v, f_u and camera center (c_v, c_u) are commonly defined as camera intrinsic parameters. They form an intrinsic matrix K as:

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Earlier K also included image skew; but by analogy with tangential distortion, it is not taken into account for modern cameras.

2.1.7 Spatial camera coordinates

Conversion from Eq. 2.5 is a generalized form of the 3D-2D point transformation. This only works if the center of camera coordinate system (CCS) is the same as WCS. Since this is not the case in the real world, the camera pose must be taken into account for this conversion.

Difference in positions between CCS and WCS can be expressed by two components, which are rotation R and translation t. Rotation R is expressed as a set of angles between the basis vectors of CCS and WCS, *i.e.* Euler angles (ψ, θ, ϕ) for X, Y, Z axes respectively. Often it formed as a *rotation matrix*, which is obtained from Euler angles as:

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix}, R_Y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}, R_Z = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$R = R_Z R_Y R_X = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.8}$$

Physical distance between origins of WCS and CCS gives a transition vector t, expressed in three coordinates as:

$$t = \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix} \tag{2.9}$$

Based on Eq. 2.8 and 2.9, the translation of a 3D point from WCS to CCS expressed as:

$$\begin{aligned} X &= \hat{X} r_{11} + \hat{Y} r_{12} + \hat{Z} r_{13} + t_X \\ Y &= \hat{X} r_{21} + \hat{Y} r_{22} + \hat{Z} r_{23} + t_Y \\ Z &= \hat{X} r_{31} + \hat{Y} r_{32} + \hat{Z} r_{33} + t_Z \end{aligned} \tag{2.10}$$

where $[X, Y, Z]$ is the 3D point, brought from WCS to CCS, $[\hat{X}, \hat{Y}, \hat{Z}]$ is a 3D point in WCS. In literature, R and t are called "extrinsic parameters".

2.1.8 Matrix form

It is much more convenient to project a point from 3D to 2D using matrix multiplication. To do this, the original 3D point coordinates are converted into so-called homogeneous coordinates by creating a new dimension for the point (making it four-dimensional (4D)) with a value of 1. Similar operation is done on 2D image point, making at 3D and creating the conditions for matrix multiplication.

Based on all information we know about the camera intrinsic and extrinsic values, the conversion can be done as:

$$\begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{bmatrix} \tag{2.11}$$

For convenience, intrinsic and extrinsic parameters are often presented in a form of projection matrix P:

$$P = K \begin{bmatrix} R & t \end{bmatrix}, \tag{2.12}$$

which allows further simplification of notation to:

$$m_d = \begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix}, \hat{M} = \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ 1 \end{bmatrix}, m_d = P\hat{M} \tag{2.13}$$

2.1.9 Distortion model

The resulting 2D points from Eq. 2.11 are still suffering from the lens distortions. For their processing these distortions need to be compensated.

A widely used lens distortion representation is a Brown–Conrady model, proposed in [23]. There, distortion of lenses are modeled as:

$$r = \sqrt{(u_d - c_u)^2 + (v_d - c_v)^2} \tag{2.14}$$

$$\begin{aligned} u &= c_u + (u_d - c_u) \sum_{i=1}^{n} k_i r^{2i} \\ v &= c_v + (v_d - c_v) \sum_{i=1}^{n} k_i r^{2i}, \end{aligned} \tag{2.15}$$

where k_i stands for the distortion coefficients, n is a number of coefficients, typically up to six, u_d and v_d are the original (thus, distorted) image points, and u, v are corresponding undistorted points, which are interesting for us, since they are used for the depth estimation routines.

2.1.10 Camera calibration

To estimate the intrinsic parameters of cameras and define their relative positions in WCS we need to do a procedure called camera calibration.

The idea behind it is to use a calibration target, which can be one-dimensional (1D) [185], 2D [184], or 3D [180], to establish relationships between the assumed 3D positions of calibration target and 2D projection of the target elements on the image plane. Corners extracted from the target are often used as elements for calibration, as they are relatively easy to detect in the image [45]. To reach acceptable accuracy of retrieved camera parameters, position of target elements on image need to be estimated with subpixel accuracy [36].

Calibration target needs to be captured by all cameras from different viewpoints. Practically, at least 20 images with good coverage of image space with the projections of calibration target are used. A constraint of one fixed object needs to be preserved; that is, either camera is moving around a static pattern, or the pattern is moved around fixed camera.

The extrinsic parameters of each camera can be determined by observing the same calibration pattern in two or more views, assuming that the 3D coordinates of the calibration target are known. In order to do this, the intrinsic parameters of each camera must first be estimated. These intrinsic parameters are then used to calculate the values of the cameras' rotations and translations based on the reprojection of the target elements. Fortunately, the intrinsic and extrinsic parameters of multi-view cameras, such as a LF camera, can be estimated simultaneously with one dataset of calibration target capturings.

For the both cases of camera parameters estimation it is solved as an optimization problem by using a reprojection error as a criterion. It measures

the difference between original image points and the projections, obtained by estimated parameters using Eq. 2.13. Hence, for the case of intrinsic parameters estimation, the previously mentioned lens distortion model needs to be taken into consideration (Eq. 2.15).

Auto-calibration

Calibration can be also done without any particular calibration target. This process is called auto-calibration. It is based on extraction of image features from the arbitrary scene and matching of these features in consecutive images. Generally, matching of these features allows to estimate a homography matrix, which can be used as a source of intrinsic and extrinsic parameters [46].

2.1.11 View remapping

By using the known intrinsic parameters and distortion coefficients of a camera, it is possible to correct the distortion in an image and transform it to the desired intrinsic parameters. This process is non-linear due to the nature of the distortion model, and can be computationally intensive if performed on every capturing. This can be a bottleneck for real-time applications that require fast processing.

Assuming that camera parameters do not change during image sequence capture, it is possible to precompute the locations of undistorted pixels for each view and apply this transformation to each new image as it is captured. For that, a concept of lookup table (LUT) is used. A 2D array stores the desired location of pixels, and for every frame the LUT is applied to it with values interpolation to provide subpixel accuracy for the relocated pixels. This can significantly reduce the computational burden of distortion correction in real-time applications.

2.2 Stereo Camera

One unpleasant thing about images captured with a single camera is the loss of information about the depth of the scene, or the distance from the camera to the captured objects. Fortunately, by using similar second camera, placed with certain constrains, depth information can be retrieved up to some point. Next step of LF camera derivation is related to principles of stereo cameras.

2.2.1 Definition

Stereo camera is a type of camera which consist of two camera sensors, placed with knows positions between each sensor. The concept is formulated with reference to human vision, in which two eyes are used to perceive depth. In this chapter we will be looking at a combination of two perspective cameras as a stereo camera.

In general, when creating such a setup, we try to position the single cameras in such a way that no relative rotation of one camera to another can occur (in other words, that the cameras are oriented in the same direction), and that the relative position of these cameras can only be determined by shifting in one of the X or Y axes (but not on Z axis, as this would prevent the necessary constraints for calculating depth and would only bring more troubles). Mathematically it can be expressed as:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} ; \; t = \begin{bmatrix} t_X \\ 0 \\ 0 \end{bmatrix} \text{ or } t = \begin{bmatrix} 0 \\ t_Y \\ 0 \end{bmatrix} \tag{2.16}$$

This constrains are ensured by proper camera placement and additionally secured using rectification based on calibration results.

2.2.2 Historical background

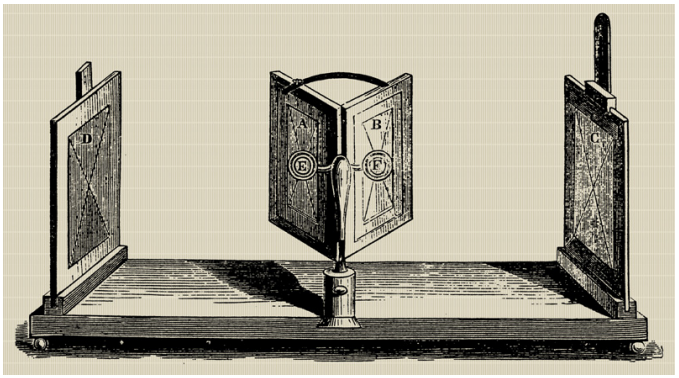


FIGURE 2.6: A drawing of stereoscope, proposed by Wheatstone in [169]. Public Domain

The theory of stereo, or binocular, image perception was described in 19^{th} century by Sir Charles Wheatstone [169]. He created a device called "stereoscope", utilizing a concept of stereopsis, or the ability of the human brain to perceive depth from two slightly displaced views of the same scene. Drawing of this device is demonstrated in Fig. 2.6 In this device, two images, captured with offset similar to distance between human eyes, were placed opposite to mirrors, inclined by 45 degrees relative to the planes on which the images are placed. While looking at the left image by the left eye and the right image by the right eye, the brain completes these images to a 3D scene, captured on the images.

Wheatstone's invention was a crucial moment in photography history as it marked the first time that a 3D image could be recorded and shown. It inspired Sir David Brewster to create an improved version of such device [168], drawing of which is demonstrated in Fig. 2.7.

In contrast to Wheatstone's, the new stereoscope was using lenses instead of mirrors to focus the images to the eyes. Such principle allowed to create

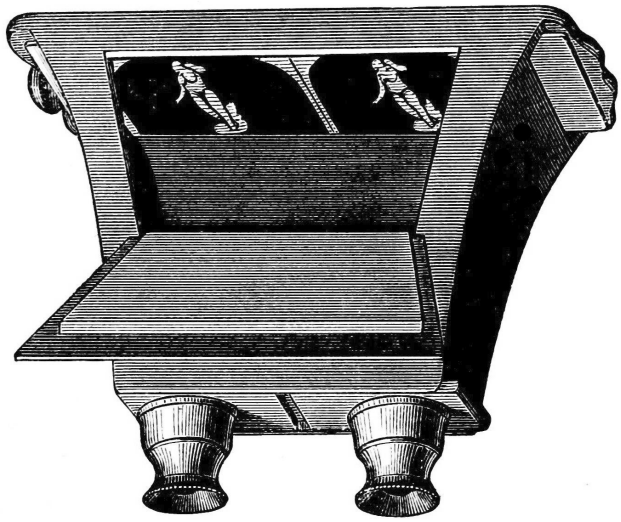


FIGURE 2.7: A drawing of portable stereoscope, developed by Brewster. Public Domain

hand-held devices, which were massively produced, and became a motivation to create stereo cards of various scenes (see Fig. 2.8), which could be inserted to and viewed by these devices. Further development of image-

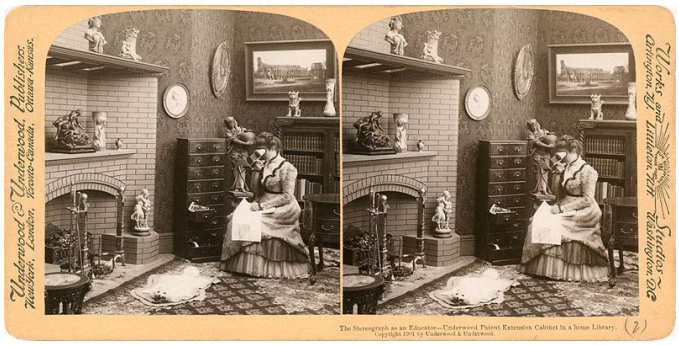


FIGURE 2.8: A stereo card for stereoscope. Public Domain

capturing devices allowed to create stereo cameras. Nowadays, stereo camera systems are manufactured by various companies around the world.

2.2.3 Parallax

As said before, stereo cameras can retrieve depth information from the scene, based on the relative shift of objects in images from two cameras. It is possible by using the concept of parallax. Parallax refers to the relative shift in the position of an object when the object is captured from two different viewpoints. By comparing the images from the different viewpoints, it can be observed that objects closer to the cameras will have a greater apparent shift in position than objects further away.

2.2.4 Stereo calibration

For the proper estimation of depth information, the cameras that comprise the stereo system generally need to have common focal length and principal point. For that, steps described in Section 2.1.10 need to applied to both cameras in stereo system. It creates a search constraint, which will be explained in the following section. Proper calibration can even compensate for the case where the camera is slightly offset from the axis on which in theory there should be no offset (i.e. for a slight vertical offset for the horizontal camera placement, or vice versa).

2.2.5 Triangulation

If we imagine a 3D point M on the scene, captured by two cameras with a one-direction shift between them, this point will be captured on image planes of these cameras as m_1 and m_2 respectively. Considering two cameras with horizontal placement, for the properly rectified cameras with the same focal length and principal point projection of the point M will lie on the same axis in both images. Due to the parallax effect, projection m_1 will be shifted from m_2 by a certain number of pixels d. This displacement is usually defined as *disparity*. The disparity of every pixel in the scene is inversely proportional to the distance from the camera system to the corresponding point in the 3D scene (*depth*):

$$Z = \frac{fb}{d} \tag{2.17}$$

where Z is the actual distance to the point, f is the rectified camera focal length (in pixels), and b is the baseline (in metric units). Using the distance information, we can retrieve the original position of the 3D point, changing some terms of the Eq. 2.1 as:

$$Y = \frac{Z\bar{y}}{f},\ X = \frac{Z\bar{x}}{f} \tag{2.18}$$

Fig. 2.9 visualizes the relation between 2D and 3D points *w.r.t.* stereo camera system.

2.2.6 Advantages and disadvantages

Stereo camera systems are the first solution that comes to mind when we talk about the task of 3D reconstruction. Compared to other systems, they are relatively simple, cheap to build and do not require much processing power for their operation. The reconstruction quality of stereo cameras can be considered suitable for simple tasks.

However, there are several aspects to consider. Having only two viewpoints may not always correctly solve the problem of ambiguity in the reconstruction, which may, for example, appear in repeating textures. Same stands for occlusions: if part of the scene is visible in one view but covered

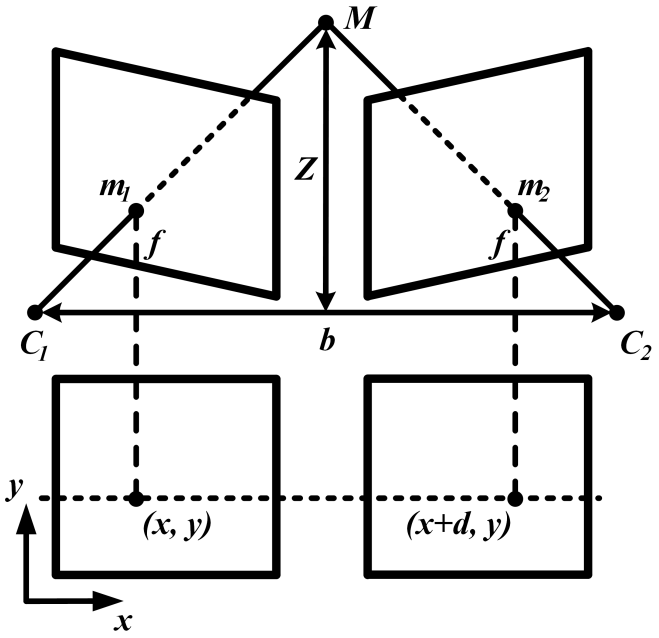


FIGURE 2.9: Triangulation of a 3D point from its 2D projections [46].

by another scene object in the other view, it can't be reconstructed. Also, it is relatively difficult to obtain disparity measurements with subpixel accuracy using information from only two images [102]. To handle that, we can use more than two views for the reconstruction.

2.2.7 Multi-view stereo

Usage of more than two images for the reconstruction using only different positions of capturing devices can be framed to the definition of multiple-baseline stereo problem [114]. Such systems are formed by placing the multiple identical capturing devices at the same distance between each other. Data about the positions of scene points obtained from many devices is accurate enough to precisely reconstruct the scene in three dimensions. Fig. 2.10 visualizes the idea of point observation from multiple cameras.

Similar to stereo case, for multi-view cameras the points projections will have certain disparity *w.r.t.* to each other views. Triangulation of a point for this case can be extended by involving all points projection to image space by using the projection matrix concept using the method of direct linear transform [46]. Additional details regarding the connection of this concept with LFs are provided in the next section.

2.3 Light Field

When we say "light", we can mean two concepts. First of all, light is an electromagnetic wave of a certain spectrum. If we mean visible light, *i.e.* light

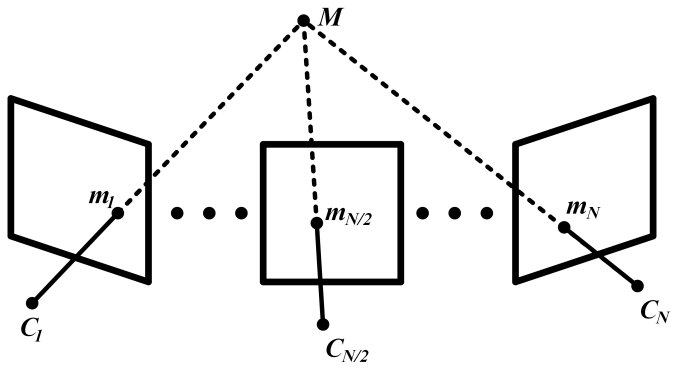


FIGURE 2.10: Multi-view visualization of a point projection.

perceived by the human eye without additional devices, these are wavelengths of 380-780 nm. But also, light can be defined as a particle, in this case a photon. The energy of a photon is inversely proportional to the wavelength and is related to it by Planck's constant. In the scientific literature, this concept is called wave-particle dualism [41].

With this information we can describe the intensity (*i.e.* color), which is carried by a certain photon. If we want to describe the direction of flight of a photon, we need five parameters for this. In 3D space, three spatial coordinates are needed to determine the point from which a photon is emitted. The orientation in which the photon was released is determined by two angles. The five-dimensional (5D) function, describing the light direction, is called "plenoptic function" [3]. Visualization of this function is shown on Fig. 2.11.

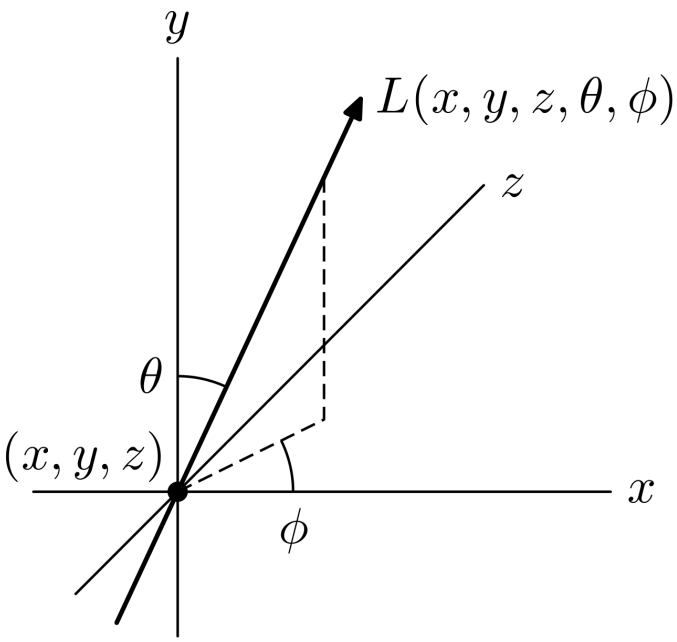


FIGURE 2.11: Visualization of the plenoptic function. Public domain

2.3.1 Definition

The plenoptic function as a means of describing finite sampling of the light was defined in the literature in 1991. However, the original definition of the LF as an infinite set of vectors determining the direction of light was given by Gershun in [38]. By him, the LF was defined as a combination of light rays, passing in every direction through all points in space.

This concept, due to its complexity, is difficult to use in computer vision tasks. Fortunately, a friendlier concept has been formulated by researchers.

2.3.2 Parameterization

Levoy and Hanrahan [86] proposed a method of parameterizing the LF that helps in representing the LF in a way that is more comfortable for the computer vision tasks. It is based on the assumption that the light ray's intensity will remain constant throughout. Hence, the 5D representation from plenoptic function can be simplified to 4D. Authors came up with the concept of representing a light ray by its intersection with two planes after analyzing various means.

On its way, the ray intersects a plane of angular coordinates (s, t), which defines the viewpoint, and a plane of spatial coordinates (u, v), which describes a view in LF; out of which the whole LF space can be denoted as L, and a particular ray is:

$$L(u, v, s, t) \tag{2.19}$$

Within this dissertation to represent different "layers" of matrices, such as LF, we will use the concept of slicing, similar to that present in some programming languages for slicing certain layers of a matrix. So, if we want to depict a two-dimensional matrix for a particular LF viewpoint we use:

$$L(\ ,\ , s, t), s \in S, t \in T, \tag{2.20}$$

where S and T stand for number of horizontal and vertical LF views. Here and further in the dissertation, by "view" we mean "2D planar grayscale or RGB image, defined by (s, t) coordinates". Fig. 2.12 demonstrates how the light ray intersects the planes, forming its location on themselves.

This representation has the advantage, according to Levoy and Hanrahan, of simplifying geometry. It is difficult to dispute this because, as will be demonstrated later, using two planes for the LF actually simplifies the computations. Note that one dimension of the parameterized LF does not have to be present, downgrading a 4D LF to 3D.

There are alternative ways of parameterizing the LF, *e.g.* based on the use of a sphere, such as the intersection of light rays with the surface of a sphere [186]; however, they are out of scope of this dissertation.

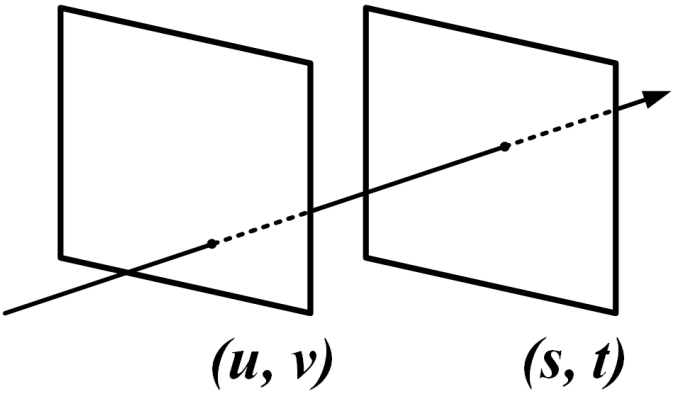


FIGURE 2.12: Visualization of two-plane parameterization of LF, proposed by Levoy and Hanrahan in [86]

2.3.3 Connection with multi-view stereo

Fig. 2.13 shows how the particular LF views can be arranged. Parameterization by the above method gives us an important property about the location of views in the LF. Using the two-plane parameterization, it is easy to observe that LF can be considered as a rigid case of multi-view setup. Views of the LF are proportionally arranged in space, *i.e.* the physical distance between the views in both dimensions is the same. Hence, we can define the LF as a set of 2D images, containing the projections of rays, and placed with same distance between every view on both axes. That allows us to consider

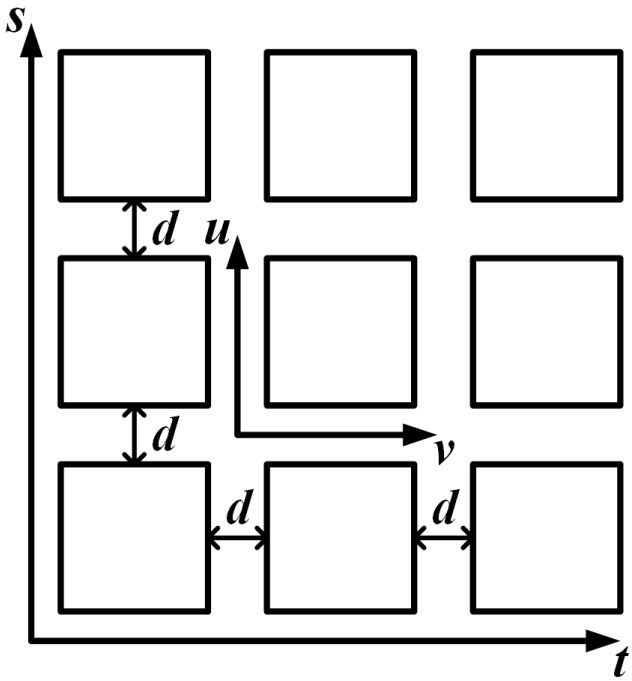


FIGURE 2.13: Formulation of a LF as a set of 2D images based on two-plane parameterization

LFs as an expansion of the multi-view stereo capturing systems, which will be used as fundamental constrain to achieve real-time performance of depth estimation algorithm.

2.3.4 Duality

An interesting property coming from the parameterization model of the LF is its duality, *i.e.* roles of angular and spatial planes can be changed to get two different LF structures [27]. In the original definition, LF is represented as a set of light rays, projected from (u, v) plane to (s, t) plane, organizing the images in an array manner. But also, the LF can be viewed from the plane (s, t) position, when rays are projected on (u, v) plane. This concept is vital for forming the alternative way of LF capturing devices design, as shown in the following section. Fig. 2.14 demonstrates how the LF takes from different viewpoints look like.

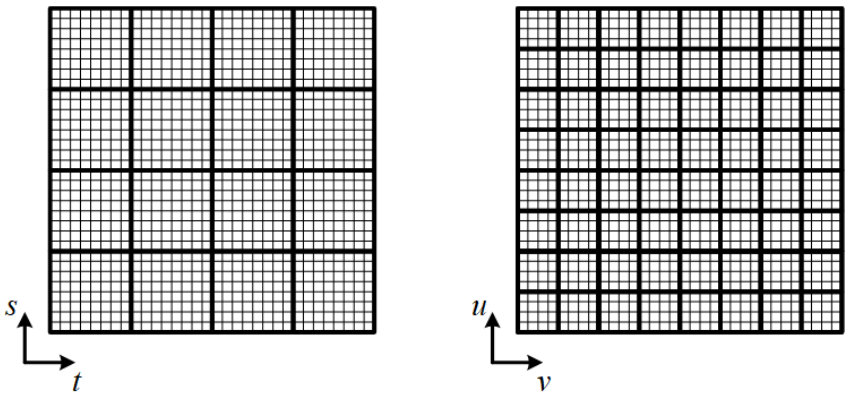


FIGURE 2.14: Visualization of the LF duality

2.3.5 Acquisition

There are different ways for capturing the LF. In terms of construction the simplest approach is by moving the single camera on the certain (and equal) distances, and capturing frames from these viewpoints. But simplicity here bring problems.

First, camera should be moved very accurately to preserve the equal baseline constraint. It can be done either by using some kind of a moving stage, where the movements are performed by electric motor, or by using additional control pattern. A method to partially overcome this issue is demonstrated on the Fig. 2.15 from [81]. There, the control pattern photo is taken by the front camera of the smartphone in the same time as rear camera captures LF frames. Knowledge of the relative position of capturing device at the time of capturing helps reducing the misplacement effect. Nevertheless, such approach is not suitable for the dynamic scenes, where the time of scene object movement most likely will be one or two order of magnitudes smaller than the camera movement time. Fig. 2.14 demonstrates how the LF taken from different viewpoints look like.

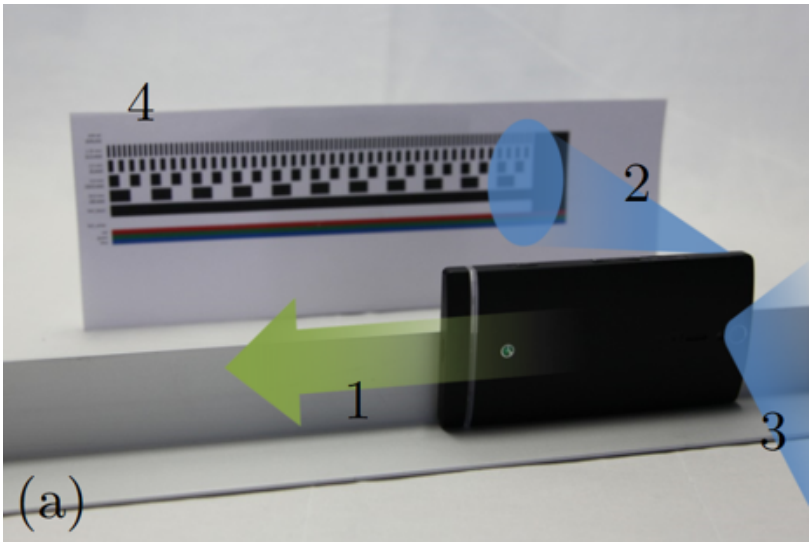


FIGURE 2.15: Demonstration of LF capturing using moving stage based on the smartphone [81]

A logical solution to the problem of capturing dynamic LFs is to simultaneously capture all views of the LF. These are two approaches of doing so. First, individual single cameras with isolated lenses can be assembled to the camera array by placing them in a one- or two-dimensional grid. An example of such configuration is shown on Fig. 2.16.

FIGURE 2.16: A LF capturing device, consisting of five independent single RGB cameras

While being able to capture the dynamics in the scene, this type of LF devices suffer from another set of problems. The main issue of such a setup is the synchronization between different cameras of the array. All frames need to be captured simultaneously, which is hard to supply without additional devices for hardware-based shutter triggering, *i.e.* capturing on the moment of electric impulse detection. The problem is that not all cameras have this functionality, and for many consumer (and even industrial) cameras that could potentially be made into a LF device, the only option for simultaneous shutter triggering is so-called software triggering, where the shutter is triggered by receiving a command through the camera application programming interface (API). Another problem of such devices can be a current spike at the moment of triggering of several cameras, which must also be considered during the design of such devices by including additional filtering elements in the power circuit and using special power supplies to

maintain operation during spikes, which in turn leads to an increase in the cost of the final device. Additionally, it is hard to assemble relatively small device using such a configuration.

All of the previously mentioned problems can be circumvented by using a more elaborate design for the LF camera. Instead of joining individual cameras into array, one camera with relatively high resolution can be used as a basis. Instead of the conventional lens, an array of smaller fixed lenses can be installed in front of the image sensor. In this case, effective resolution of every frame will depend on the original sensor resolution and number of installed lenses. All LF views are captured simultaneously, overcoming the issue of synchronization; in the same time such a device is simpler in manufacturing.

FIGURE 2.17: A LF capturing device, consisting of single RGB camera with 4x4 lens array in front of the image sensor

An example of the camera of this type is shown of Fig. 2.17. It consists of a camera sensor with 4×4 lens array in front. Early prototypes of such devices were described in 1960s [110], similar devices in smaller form-factor aimed for smartphone usage were described and produced by Pelican Imaging in 2013 [158]. This configuration also imposes its own limitations on the size of the device (*e.g.* it will be difficult to take high-resolution photos with

such a device), but for most possible applications of such a camera the image resolution should be sufficient.

What is interesting is that the concept of LF capturing by the array can be "traced" to 17^{th} century, as demonstrated on Fig. 2.18.

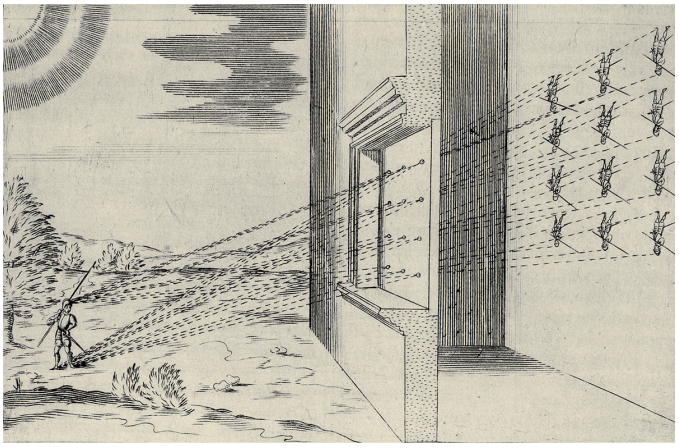


FIGURE 2.18: Illustration of a twelve-hole camera obscura from Bettini's Apiaria universae philosophiae mathematicae [16]. Public Domain

Another limitation of such a configuration is related to the device size. It is hard to make relatively small LF camera with full-size lenses. Fortunately, LF duality principle, described in Section 2.3.4 allows to make LF capturing device by using a big number of small lenses. In other words, instead of $N \times N$ big lenses capturing $M \times M$ pixels in each view the $M \times M$ array of small lenses placed in front of camera sensor provides $N \times N$ small images.

This concept is known in the literature as "plenoptic camera" [4]. In general, cameras of this type, in addition to the main lens, are equipped with an additional lens on which there are small lenses (hence often called Micro-lens array (MLA)). Most of the LF cameras produced operate according to this principle. An example is the handheld camera produced in the past by Lytro [111]. However, MLA principle is not limited to the usage of two lenses. An interesting design utilizing optical fiber as light-receipting device for LF capturing was proposed by Orth *et al.* [117]. Another notable example of modern device of this type is inspired by the vision system of extinct trilobite [33].

While being one of the most popular ways of designing LF cameras, due to the complexity of semi-homemade test cameras of this type and the unavailability of affordable solutions on the broad market, cameras of this type are out of scope of this dissertation.

2.4 Other concepts

This section will define the concepts necessary to explain the topic of this dissertation.

2.4.1 Depth Estimation

Depth is one of the forms of the 3D reconstruction results representation. In general, it represents a mapping between 2D image coordinates and the distance from the camera to the corresponding point of the scene [77]. These distances are relative to the camera position at the moment of capturing.

Another connected concept to the depth map is "disparity map" In contrast to depth maps, for every image pixel disparity maps store the information about the pixel shift in image coordinates. It is often utilized as and is subject to conversion to depth, since disparities map are reverse proportional to depth, as shown by Eq. 2.17.

There are other ways to represent 3D data, such as:

- Point cloud – a set of 3D points, usually in metric coordinates, representing the actual points positions on the scene. It can be obtained by depth map post-processing.
- Mesh – a set of vertices (*i.e.* point cloud) with edges and associated polygons, which together form the shape of the object. Can be generated from point cloud by its triangulation.
- Voxels – volumetric pixels grid, consisting of cubic cells, which store the information about the scene point.

2.4.2 Embedded Hardware

Embedded Hardware usually refers to computer hardware that performs a specific and limited function by being embedded in other hardware [48]. Such systems are themselves essential components of modern electronic devices and are present in many of them [10].

A number of limitations inherent in such systems follow from this definition. The main limitation is related to the power consumption of such systems, which it is desirable to keep as low as possible. This results in limiting the computational power of such hardware as an obvious way to reduce power consumption.

Real-time Requirements

Many usage scenarios involve computation in the real-time domain. In general, real time refers to the time constraints on the execution of the algorithm. Typically, for computer vision algorithm this constraint is usually set to be equal or more than 30 frames per second (FPS). However, it should be higher for the critical applications, *i.e.* autonomous driving.

2.5 Conclusion

In this chapter the main ideas on which this thesis is based were presented. We showed how the concept of basic single camera can be expanded to stereo and multi-view cases. Ways of formulating, capturing and describing LFs are

explained, followed by its connection to the multi-view camera case. In the next chapters the literature reviews of existing algorithms for LF processing related to the main tasks of this dissertation is presented.

Survey on Related work

In this chapter we are going to discuss State-of-the-art (SoA) approaches for the estimation of LF camera parameters and the depth estimation algorithms from LFs.

3.1 Light field calibration methods

The earliest work describing calibration principles for LF cameras was published by Vaish *et al.* in [157]. The authors address the question of estimating the minimum required calibration for LFs. They argue that for some applications it is sufficient to estimate the calibration data up to affine ambiguity. Assuming that the LF images are aligned close to a common plane, they developed a framework using planar parallax to obtain the relative position of the LF views. Although this method is useful for viewpoint rendering and refocusing, the authors state that its accuracy is insufficient to use in place of metric calibration for the 3D reconstruction problem. It can be seen *e.g.* in [71], where plane and parallax-based calibration was used for the estimation of the *relative* depth maps.

Work that showed how the principles of stereo rectification can be used to align calibrated LF species was presented by Kang *et al.* in [74]. A common baseline for the replaced views is estimated iteratively using the views center positions, which minimizes the distance for the new views placement. For aligning the views orientation to the common plane transformation with the assumption of close-to-aligned views their rotations are averaged. The intrinsic parameters are adjusted in a similar way by averaging, and the overall focal distance is chosen as the largest of all the original focal distances. The efficiency of averaging values for rectification prompted us to use a similar initialization of the views rectification parameters.

More recent work of Xu *et al.* [172] shows the end-to-end calibration and rectification routine for LF cameras. For every viewpoint its intrinsic parameters are estimated using classical pattern-based calibration [184]. The extrinsic parameters are combined to Jacobian matrix, tailored with the knowledge of views placement constrain, and optimized together using the Levenberg-Marquardt algorithm [104].

An interesting approach of LF camera calibration was presented by Zhang and Chen in [181]. Authors were using the pattern-based calibration from [20] to obtain the intrinsic camera parameters. Extrinsic parameters were

estimated using the Zhang's calibration [184]. The manner in which their LF camera is built is one of the aspects that makes their approach especially fascinating. Individual cameras in the array are mounted on two servomotors, which enables movement in two Degrees of freedom (DoF) (pan and sidestep) for every camera. Based on the camera position data, obtained from calibration, cameras are adjusted to face the common plane. However, the viability of using this method to produce small LF cameras is questionable.

In the works that have been mentioned before, the case of the conventional representation of LF camera as a collection of normal planar cameras has been taken into consideration. Up to a certain point, these approaches can be also used for calibration of array-based LF cameras due to the duality principle, described in Section 2.3.4.

In the publication of Zhou *et al.* [187] the plenoptic camera calibration problem is solved in proposed epipolar space geometrical model. In this model a concept of Epipolar plane image (EPI), which is described in Section 3.2.1, is used. Using this model, relation of 3D point to their LF projections are described through a 3D vector, which consists of the slope and intercepts of the respective line in EPIs. The calibration task is initially solved by homography-based solution using the calibration pattern and further optimized non-linearly using Levenberg-Marquardt method [104].

Ji and Wu in [68] propose a calibration model for plenoptic cameras, based on collinear constraints and utilizable with a conventional calibration target. Working on sub-aperture images, their method finds the positions of center views and use them as a reference for the rest of the views. Having this data, the position of other views is estimated based on the calibration pattern corners projection *w.r.t.* the reference views. It is done by combining the equations for the corners to the matrix form and using bundle adjustment algorithm [154] on them.

In contrast to other approaches, method of Bok *et al.* [17] use RAW-captured images instead of preprocessed ones. Using a classical checkerboard pattern, instead of corners line-alike features are extracted from sub-aperture images. Lenses distortion and misplacement inaccuracies are estimated using these features. Initial estimations are followed by non-linear optimization.

Similar to [17], Noury in [112] use the RAW images for the calibration purposes. However, they use corners as the features for calibration procedure, which are extracted from the calibration pattern. The initial camera poses are refined non-linearly.

A different feature extraction method of plenoptic cameras calibration is proposed by O'Brien *et al.* in [113]. Using the checkerboard calibration pattern authors extracts disc features instead of conventional corners for retrieving the LF calibration data.

A so-called "blind calibration" was proposed by Sun *et al.* in [146]. Authors use three calibration targets, placed proportionally on fixed distances between them. For the feature matching gradient-based correspondences search scheme is used.

In the work of Bergamasco *et al.* [13] a parameter-free camera model for the independent LF rays representation is proposed. A flat personal computer (PC) display is used as a source of dense calibration targets, generated using phase coding technique. Calibration is done based on the triangulation.

3.2 Light field depth reconstruction methods

This sections presents a broad overview of existing methods of LF depth reconstruction.

3.2.1 Classical methods

A concept of EPI was introduced by Bolles *et al.* in [18]. Their research was based purely around the question on how to extract 3D data from a given sequence of pictures and was originally formulated for solving Structure-from-Motion task. LFs were not involved in their research since the concept of LF cameras was not yet available. In their experiment Bolles *et al.* put up a camera rig on rails and moved the camera lateral to the scene. While the camera was moving it took a dense sequence of images. Stacked images can be seen as a solid block and a horizontal or vertical cut through this block reveals the EPI.

By using a two-plane parameterization of LF, described in Section 2.3.2, for a light-field $L(u, v, s, t)$ EPI can be obtained by fixing one angular and one spatial coordinate, *e.g.* (v, t), which results in a two-dimensional image of (u, s) coordinates. LF rays are projected to this image as lines. An important property of the EPI is related to the proportionality of the slope of this lines to corresponding point depth value. This representation provides a simplification for the scene analysis and therefore was used in various 3D reconstruction algorithms.

One of the most notable series of works on the EPI analysis for the depth reconstruction purposes was published by Wanner and Goldlüecke in [164]. It presents a variational framework for LF analysis with a specialized local data term for depth estimation, designed to align with the structure of LFs. This metric is more resilient compared to conventional techniques when applied to non-Lambertian objects. Their work draws from previous work done with EPIs, which have already shown the capability of EPIs for detecting edges, peaks and troughs with a subsequent line fitting in the EPI to reconstruct 3D structure.

Wanner *et al.* in [165] expands on their previous paper [164], making several enhancements. For disparity estimation, the local slope on EPIs for the two different slice directions using the structure tensor is found. In 3D space, a point is projected onto a line, where the slope of the line is related to its depth. The calculation results in two local disparity estimates for each pixel in each view. These disparity estimates can then be merged into a single disparity map in two different ways: either by locally choosing the estimate with the higher reliability, optionally smoothing the result, or by solving a global

optimization problem. The algorithm is very robust for relatively small disparity ranges; however, the results get worse for larger disparity values when impulse noise is added to the input images.

A noticeable work in EPI analysis for the depth reconstruction was published by Kim *et al.* in [75]. It uses densely captured LF from the moving stage. Authors propose to use fine-to-coarse *(sic)* strategy for the accurate reconstruction. In this strategy depth is estimated at edges in the EPI at the highest resolution, then this information is propagated throughout the EPI, and analysis continues on the downsampled EPI resolutions. For that, this method first identifies regions where the depth estimation is expected to perform well. Using the edge confidence measurement, depth is estimated for the edges with a high value of this measurement. The depth estimation is done based on intensity values comparison on the edge and its neighbors in angular EPI dimension with kernel density estimation (KDE). Depth is propagated among all pixels within the same line, similarity of which is defined by Euclidean norm. Depth estimation and propagation is iterated until all EPI-pixels with a high edge confidence have been processed. The homogeneous regions of EPI at this stage are left without the assigned depth. They are analyzed after the EPI downsampling by applying the previously described procedure of edges confidence measurement. In the examples, provided in paper, relatively big EPI images are extracted from dozens of views. In order to solve the problem of omission of part of the objects in all the captured views, the reference line is moved on the angular dimension of EPI.

An interesting feature of LFs is that it can store not only 3D information, but also information about the focus of the camera. Hence, images taken by LF camera can be refocused. With some geometrical calculations and knowledge about the lens setup it is possible to calculate to which distance the camera is focused. Thus, if an object in the image is in focus, we can determine its depth. This information is called defocus cue.

Approach of Tao *et al.* , published in [150], suggest the analysis of defocus cues alongside correspondence cues for solving the 3D reconstruction task. The reason for doing so is that defocus depth estimation performs better in repeating textures and the results are quite consistent but blurry, whereas with correspondence cues it is possible to produce sharp results even on bright features, but the result suffers from inconsistency in noisy regions. Both cues are extracted from EPI. By shearing EPIs, algorithm tries to find the shearing value which maximizes the defocus and minimizes the correspondence response. Due to possible inconsistencies of the optimal shearing value between cues, a peak ratio [53] is applied to provide certain confidence for the shearing values. These local measurements are further propagated among all depth map pixels using Markov random field (MRF) [89].

Previously described in Section 3.2.1 work of Tao *et al.* [150] was extended by Tao *et al.* in [151] by adding shading cue, which is used as a source for the lighting information, estimated by linear least squares (LLS). Authors argue that for the low-textures object using only depth and correspondence cue is not enough, since additional wrongly estimated pixels can be found for low-texture areas. Unlike the original approach, analysis of LF is done without

EPIs extraction. For each spatial pixel, a small neighborhood patch of the refocused image is compared to the corresponding patch in the center pinhole image, which forms defocus response. The correspondence response is the difference between the refocused angular pixels at the estimated depth and their respective center pixel averaged over the number of angular pixels. To reduce complexity, the minimum values of these responses are considered. A combined response is obtained by taking the weighted average of the individual response measures with their respective confidence values measured using Attainable Maximum Likelihood [59], which centers a Gaussian distribution at the minimum cost value of the particular pixel to retrieve the pixel's cost. The minimum of the combined response curve is identified as the optimal depth value for each spatial pixel. The depth constraint for each spatial pixel is computed as the squared norm of the difference between the optimized depth and the local depth estimate, weighed by the confidence measure at that position. The smoothness constraint is obtained by applying three smoothness kernels on the optimized depth. Since shading constraints depend on the entire space of spatial and angular pixels, a mapping of spatial pixel space to a combined pixel space is done for the optimized depth. The error metric for the shading term is formulated as a combination of the local and non-local shading and albedo constraints and the angular coherence constraint. The three components, depth, smoothness, and shading, are each regularized using the respective cues to give the new error metric, which is then solved for minimization using a nonlinear least squares (NLLS) approach to get the final optimized depth map.

Approach of Zhang *et al.* [183] aims on handling occlusion cases during 3D reconstruction. Due to the possible presence of noise and occlusion points in EPI the use of contour detection algorithm for the line estimation (slope of which, as said before, is proportional to depth) may lead to false estimations. Since EPI contains linear structures, points around a line can describe the line orientation. With that in mind, a parallelogram operator is defined on the segments of the EPI based on the color consistency. The orientation is found by rotating the parallelogram and maximizing the distance between the distributions of pixel values on either side of the lines. Sheng *et al.* in [136] extend the work from [183] for the multi-orientational EPIs.

Johannsen *et al.* [70] present an EPI method for depth estimation in LFs based on the sparse decomposition. Authors propose usage of LF dictionaries, which encode information regarding the depth of the scene points. First the dictionary is learned for the central view. It is utilized to code a sparse representation of the LF and then leveraged for the whole LF. Training of the dictionary is done based on the patches, extracted from EPI. Disparity is estimated by analysis of all values in dictionary, associated with every disparity level. Data and smoothness term optimizations are done for the depth estimation, while total generalized variation is used for solving the inpainting problem. It works good for the Lambertian surfaces, but for non-Lambertian comparison of two nearby depth layers is done for the proper estimation.

In the work of Tomioka *et al.* [153] the microlens LF views are dynamically

remapped based on the certain value, in a similar manner to [150]. However, instead of remapping EPIs the shifting operation is applied directly to the views. This step is done instead of introducing the disparity shift during matching cost generation. Remapped views are the subject of census transform, which makes binary strings from the pixel surrounding and will be explained in Section 5.1.1. For census-based matching cost the way how images shift can potentially provide better accuracy, since by using disparity shift on census-transformed images the subpixel accuracy is limited. Authors propose a scheme of majority operator usage for reducing the effect of noise, which sets the common bit string values as statistical mode of all values from the remapped views on the same position. Disparity values are estimated by minimizing the matching costs. Optimization of the depth map is done based on variational methods with one data term and two smoothness terms.

Work of Lee and Park [82] propose a unified model of LF depth estimation. It is done based on fusion of stereo, focus and defocus disparity estimators. Unique feature of this approach is the disparity representation using complex values. It allows to keep depth information in both cartesian and polar coordinates. Based on this representation disparity is estimated using a combination of previously listed estimators in a more convenient way compared to classical cartesian representation. More specifically, the real and imaginary parts of the complex-valued disparity are derived from the cartesian representation, and the disparity magnitude and direction are derived from the polar representation. Part of the paper is devoted to the structural analysis of LF gradients. Authors propose average gradient approach instead of previously used LLS-based method, arguing that geometry of LF can not be fully computed using LLS.

Approach of Neri *et al.* [108] uses multi-resolution scheme for the depth estimation. Motivation of this approach related to the increase of computational complexity in cases when local optimization methods are replaced with global ones. Hence, only data term optimization is used. Suggested method uses patches comparison based on Euclidean norm between reference and other LF views.

Jeon *et al.* in [66] suggest an algorithm for depth estimation from Lytro LF camera, which at that time outperformed the Lytro software approach for this task. It is based on stereo matching principles, which are extended for LF case. Here, LF views are transformed using phase shift theorem. It allows further sub-pixel matching cost generation by weighted combination of sum of absolute differences (SAD) and sum of gradient differences (SGrD). Initial depth map is estimated by minimizing a cost function. Additionally, matched SIFT [96] features are used for outlier rejection and depth map filtration. Depth is optimized by using graph cuts [78] and refined by method from [174]. This work was extended in [67] by using cascades of random forests on the various matching costs for the depth prediction, based on image similarity measurements such as SAD, SGrD and zero-mean normalized cross-correlation (ZNCC). Authors argue that for different captured scenes different combinations of matching costs give the best results. For proving

that, two random forests are constructed for solving classification and regression tasks. Classification forest determines which costs have more influence on the result, while regression forest is used for the depth values prediction from the selected costs.

In work of Yücer *et al.* [177] LF gradients are utilized for the depth reconstruction. Similar to EPI principles, depth estimation relies on angulo-spatial LF volume. However, instead of full images, patches are extracted from this volume. Several depth maps are estimated and merged with certain confidence. Filtering is done based on voxelization and depth back-projection. Depth is then propagated based on the novel edge-aware bidirectional photoconsistency measurement.

Approach of Strecke *et al.* [143] uses a peculiarity of LF. Integration of LF volume along different orientations can provide views with different focus planes. The algorithm is based on the fact that for every depth map pixel there is a symmetry in the focal stack around the ground truth disparity. Cost is built on the partial focal stacks (which are done as occlusion-free) with the robust distance function by comparing the values on the same expected depth within four partial stacks. Unlike most of the algorithms, this method also extracts surface normals from the focal stack. This data is used jointly with depth to improve the quality of the latter by optimizing both modalities together. As a small drawback, the method gets a slightly reduced noise resiliency, as it operates only on a crosshair of views around the reference view as opposed to the full LF.

Hou and Jung in [58] propose an occlusion-aware scheme for the LF depth estimation. Authors use support vector machine (SVM) for the classification of occluded pixels on refocused images. It is based on the observation that the photoconsistency assumption is satisfied for pixels that are not occluded, but not for those that are occluded. Classification is done using the difference between pixel values in original center and refocused LF views. For the pixel-wise depth estimation occluded and non-occluded pixels are treated separately. To improve the quality of the result, authors propose a segmentation-based bilateral filtering post-processing step, where depth pixel surroundings are checked for consistency to prevent blurring on edges.

3.2.2 Deep Learning-based methods

Interest in deep learning technologies and their use in computer vision applications has increased in recent years. This can be explained by the fact that the computational capacity of systems supporting parallelism has greatly increased, which allows to perform a greater number of operations on the data per unit time. For certain datasets we can say that the result generated by algorithms based on deep learning will be more accurate than the results of classical algorithms, which can be indirectly confirmed by analyzing the results of different groups of algorithms on [1].

However, at the moment these technologies can generally not be used for real-time 3D reconstruction of LFs on embedded hardware due to the high

demands of the algorithms on the hardware specifications, and consequently on the power consumption.

The fact that it is typically more difficult to obtain a generalized version of the deep learning algorithm that can be used for any camera imposes its own limitations. While obtaining such data is not difficult for synthetic datasets, it provides some complexity for real-world datasets, as it requires the accurate source of the ground truth. In general, algorithms for 3D reconstruction that use deep learning are trained using supervised learning, where ground truth, i.e. reliable information about what the 3D scene actually looks like, is required to determine the depth dependency of the scene on the captured representation of that scene.

One of the most popular types of neural networks used for image analysis and 3D reconstruction in particular is Convolutional neural network (CNN) Heber and Poke's [49] work uses CNN to estimate depth from LF data. Using the proposed network, the reconstruction problem is formulated as a prediction of the orientation of the 2D hyperplane associated with each light ray projection in LF. The orientation of this hyperplane correlates with the depth value for each scene point. In the consequential work, Heber *et al.* [50] extract the features from LF by CNNs. Authors extract 3D subvolumes from the LF, which are the subject of analysis by 3D convolutions. Network is constructed in encoder-decoder manner with additional skip connections.

Sun *et al.* in [145] describes how depth-related features can be extracted from the EPI. It uses information about EPI edges, which are considered as borders of homogeneous areas in the LF subset. For the processing EPIs are enhanced by normalization and edges localization. Additionally, Radon and Hough transforms are applied to the enhanced EPIs for the reparameterization. Similar to previously described methods, CNN is used for the depth reconstitution.

In method of Luo *et al.* [98] depth estimation is done on EPI-patches, processed by CNN. The use of patches instead of full images is justified by the desire to reduce the amount of information used by the network. Patches are selected by the presence of edge information, which is verified using Canny edge detector [25]. For every detected edge a pair of patches from horizontal and vertical EPI is formed. Two separated and identical network branches are used for the features learning from horizontal and vertical patches. Outputs of the CNN are the subject of global regularization with data, smoothness and label terms.

In their work [170] Williem and Park are referring to graph cuts [21] as to a basis of the presented approach, which specifically aims on analysis of the noisy LFs. They expand it using the LF properties, in particular availability of angular resolution. Authors propose two new matching costs, which analyze correspondence and defocus cues. For the first cue authors use the probabilistic angular entropy metric, which analyses randomness of photo-consistency within the angular patch. Defocus cue is designed in an adaptive way, which not only provides the proper information on noisy areas, but can also handle occlusions. Having a region extracted from LF it is divided to subregions, in which the presence or absence of blurring is checked, and the

response for the region is formed as the minimum of the response from all subregions. It also involves additional color constraint to reduce ambiguity between occluder and occluded region. This work was extended by Williem *et al.* in [171]. Authors propose two novel data costs for LF depth estimation: constrained angular entropy cost is measured using the angular patch, in which the pixels are weighted based on color similarity; and constrained adaptive defocus cost, which is estimated on refocused images.

Approach of [149] aims on the reconstruction of glossy, or non-Lambertian [80] surfaces. They built the algorithm based on the dependency of color intensity from the light source position and color, which does not remain constant between the LF views. Using the dichromatic model, the color of light can be distinguished from the surface color [135]. Views of LF can be refocused, which helps to analyse the additional light information. Based on that specular-diffuse separation can be done. First, the depth information is estimated from the images as-is. Depth is used for estimation of the light source color based on LF refocusing. This information is used for the separation of specular-free views from original ones, which are used for the depth refinement and LF update. The procedure is done iteratively till there is no change in the LF, and the result is refined using MRF.

A framework for depth estimation from LFs, presented by Wang *et al.* in [160], aims to handle occlusions by utilizing photo-consistency measurements in angular resolution of LF. It is based on the assumption that having an angular patch, occluded by the same occluder in all views, the patch can be divided to two sub-regions with and without occlusion, in which the photo-consistency of unoccluded area will remain same after refocusing to occlusion plane. The method uses previously discussed method from Tao *et al.* [150] as a basis for depth estimation. To find possible occlusions candidates an edge detection algorithm [25] is first applied to the angular patch. Refocus to possible depth values is done by shearing of LF data for every pixel on the detected edge. In case when occlusion is not present, variance in the patch compared to the reference view is small. For occluded regions the assumption of photo-consistency does not hold. Such regions are divided to two parts, and part with the least variance used in computations of defocus and correspondence cues. Regularization of depth is done using graph cuts [21].

Method of Liu *et al.* [93] proposed a stereo derivation of LF depth estimation process. There, LF captured by microlens-based camera are the subject of rendering enhancement, where color and vignetting correction are applied to the LF together with bilateral filter-based denoising. Authors use the stereo-derived process by analyzing image similarities and building matching cost using the locally-scaled sum of squared differences (SSD) on sub-aperture intensities and gradients for the views pairs. Depth maps, obtained by this method, are filtered by left-right consistency check and fused by weighted median filtering. Refinement of the resulting depth map is done by smoothing method from [174].

In search for the optimal depth estimation method from LF videos Dabala

et al. in [31] proposed multi-resolution approach. There, LF images are downsampled using Gaussian pyramid in a similar to Kanade–Lucas–Tomasi feature tracker (KLT) [97] manner. Matching cost generation is done by analyzing view pairs using Hamming distance measurements on census-transformed images. Depth and confidence maps are estimated independently on every pyramid level and further filtered by a consolidation procedure, where per-view confidences are merged in a weighted manner and used to discard invalid pixels. Depth maps are then upsampled to be used as the initialization on the higher pyramid level.

In work of Sabater *et al.* [131] the problem of analyzing the LFs captured by separate cameras of the same model, the color characteristic of the images of which may differ from each other, is touched upon. To cope with that, the color homogenization step is performed based on black level measurements, providing bias and gain maps. For bringing the LF views to the same plane authors suggest a warping-based scheme for pseudo-rectification, which are projected from the original views to the reference. Correspondence matching is done based on ZNCC and exploiting a multi-resolution strategy, similar to [31].

In the work of Huang [61] an empirical framework based on MRFs was developed. Global energy function for MRF is formed based on data and smoothness terms. Author is using pseudo-likelihood approximation [15] for the terms, where data component relies on view-wise neighborhood, while spatial neighborhood forms the smoothness component. Disparity map is estimated by optimizing this energy function.

Method of Lin *et al.* [92] is based on the symmetry analysis of the LF focal stack, which is a sequence of LF images with a certain focus. Authors exploiting the fact that for non-occluded pixels in a focal stack (which can be either produced by capturing the LF with different focus or be synthesized using LF properties) a symmetry exist along the depth of focus dimension. Also, a data consistency measure derived on focal stack synthesized from estimated depth is proposed. The method is modeled using MRFs with data and smoothness terms.

Qin *et al.* [124] propose usage of consistency metric range tensors for the depth estimation task. By using the fact that angular patch from LF is photoconsistent if the reference pixels are not occluded method analyzes the tensor information, extracted from sheared LFs, by measuring the variance of patches on various depth labels. Authors propose additional confidence analysis to reject wrong depth estimation. Depth is further propagated by exploiting the constrain of neighbor pixels depth similarity based on color consistency. This method if further optimized by authors for GPUs in [123].

The division of depth maps into foreground and background *w.r.t.* focus plane is the basis of the method proposed by Lee *et al.* in [83]. Gradient information, obtained by Sobel operator, is used to create a constraint for the voting decision of belonging of the particular angular patch to foreground or background. Authors propose to accumulate binary maps, obtained from foreground and background images, for the memory efficient depth estimation.

Some methods use optical flow reformulation [56] for the LF depth estimation task. Zhou *et al.* [188] build a LF flow model on the pixels displacements *w.r.t.* reference view. Authors argue that any two neighbor views of LF contain a flow and the approach tries to recognize the flow from all planes using phase shift theorem, similar to previously mentioned approach of Jeon *et al.* [66]. To take occlusions into consideration, pixels are verified for the presence in both original and centro-symmetric views.

Chen *et al.* in [29] measuring displacements from optical flow and converting them to actual pixel displacements. For one angular dimension a 3D volume is extracted by combining all LF views on this dimension. Authors propose usage of coarse-to-fine matching method on angular patches as a source of initial flow obtained by minimization of the matching cost. This flow is a subject of edge-aware feature flow filter together with weighting by a confidence map. Flow-to-depth reformulation is done by optimization on data and smoothness terms [128].

Work of Navarro and Buades [106] uses optical flow for the depth values interpolation. They use a coarse-to-fine strategy of stereo estimation on multiple scales for several view pairs of LF with further fusion of the result. The optical flow estimation is enhanced by introducing fidelity term, which verifies how close the estimated flow is to the preestimated disparity.

Shin *et al.* in [138] presented a network, which works on LF subsets. Subsets are extracted from horizontal, vertical and diagonal angular direction of LF *w.r.t.* reference view. These subsets are treated separately by identical CNNs and the generated features are merged by a similar network. Such a strategy also allows to use augmentation by shifting the reference view and extracting different subsets.

Further development of attention-based method was done by Chen *et al.* in [28]. There, attention module is based on information regarding scene occlusions and divided to two branches. First combines features based on relative amount of the occlusions in the LF views, while second aims on interchanging of information between combined features. It is done to provide higher contribution of the occlusion-less information to the matching cost.

Zhu *et al.* [190] proposed a depth estimation algorithm that is robust to multi-occluder occlusion by exploring the occlusion point and non-occlusion point. They modeled the multi-occluder occlusion and defined the occluder-consistency property that corresponds the views of the occluder and background in spatial patch to the occluded view and un-occluded view in angular space.

In the work of Leistner *et al.* [84] a method called EPI-Shift is proposed. To operate on a wider range of disparity, EPI-Shift utilizes the concept of epipolar lines in image space to represent depth shift. The method divides depth estimation into classification and regression tasks, with each shift considered as a discrete disparity label. A sub-CNN extracts local disparity information from each resulting image stack, and a U-Net architecture integrates and concatenates this information to obtain a discrete depth label and a continuous sub-pixel disparity map. The final result for each pixel is the sum of the highest classification based on shift and the corresponding regression value.

Another paper by Leistner *et al.* [85] proposes methods for handling semi-transparent regions in depth estimation, and introduces a multimodal LFs depth dataset to handle overlapping objects at different depths. If is based on three deep learning-driven approaches. The Unimodal Posterior Regression approach tweaks the loss function of the Maximum Likelihood learning model by using an extended version of the $L1$-loss function with a variable width, to measure uncertainty in a pixel. The EPI-Shift-Ensemble method uses an ensemble of networks and takes advantage of the nature of LF images to shift the input and output multiple times. The Discrete Posterior Prediction method discretizes the output of disparities and uses the softmax function to represent the posterior, which can result in overconfident but incorrect predictions.

The paper of Chen *et al.* [28] proposes an attention-based multi-level fusion network. This method analyses the line structures from EPIs. Occlusion varies within and between EPIs, so attention is used for proper EPI selection. Attention-based intra-branch and inter-branch features fusion is used to select EPI directions with less occlusion.

The paper by Huang *et al.* [62] introduces a lightweight disparity estimation model. It works by selecting only the sub-images along the horizontal and vertical directions and grouping them as anchors based on their angular distance from the center view. Paper introduces bottleneck attention module and works on multi-disparity-scale cost aggregation with edge feature extraction for the edge guidance to produce better results in regions with fine and detailed structures.

Tsai *et al.* [156] proposed an attention-based approach, where specific weights are assigned to every LF view based on its placement to avoid redundancy in the information for depth map estimation from LF images, achieving a good balance between accuracy and computational complexity. Their method generates an attention map for each LF view based on its subjective importance for the reconstruction process. This attention map is used to scale the features, preserving information from the views with significant contributions.

Unsupervised methods

Most of the unsupervised depth estimation algorithms relies on two paradigms. To a greater extent the core of the unsupervised methods lies in the specific design of the loss function, which allows to somehow reflect the relationships between the LF views and the estimated disparity map. Less attention is paid to the design of the neural network itself, because suitable loss function contributes more decisively to the estimation result. Thus, we can conclude that a properly engineered loss function can be coupled to any existing algorithm for disparity map estimation, which we will take advantage of.

In this way, the algorithm presented in the work of Iwatsiki *et al.* [65] works. Using the popular supervised method EPINET [138], instead of redesigning it, the authors propose a general loss function design based on projections of the LF views into the coordinates of the reference view. After

warping the set of images to this view, authors propose averaging of the images to prevent the noise effect. Our work is using slightly different strategy for dealing with noise by proposing a per-view mask for the warping errors filtering, which will be explained in the corresponding chapter.

The first method for unsupervised LF depth estimation was proposed by Peng *et al.* [120]. There, the idea of LF views warping to the reference view coordinates was presented. Loss function is constructed on the by analyzing the similarity of these warped views with reference one by Euclidean distance and in the same time the similarity between all warped views within them by the same metric. As a reference view, four corner LF views are used, and the loss is estimated by combining the estimations for the different references. For dealing with mismatched values during patch-based training, authors propose optimized warping strategy by padding and consecutive cropping of the images which are subject to warping. In their consecutive work [121] the approach was reformulated to become zero-shot method while keeping the same principles of loss function design.

Li *et al.* in [87] design a framework for unsupervised depth estimation from EPIs. They form stacks of EPIs, which are obtained from deformed LF views and learned by neural network to form the shared weights.

The work of Jin and Hou [69] paid additional attention to the problem of dealing with image occlusions and propose the occlusion-aware model. It is based on the assumption that for a particular angular patch, after its division on four equal parts, at least one of its parts should be consistent with the reference and hence don't store any occlusions. Authors use the sub LFs instead of original ones, which are generated by that, the sub-LFs are used instead of the full ones. Each subset produces so-called uncertainty-aware disparity maps, which are optimized individually by the loss function and further fused based on the reliability maps.

In the work of Zhou *et al.* [189] LF views are extracted in the "star"-like manner, and analyzed separately as the central sub-LF and corner views, namely inter and outer views. They argue that proposed utilization of outer views helps to prevent vanishing gradients issue during unsupervising training. Authors propose usage of three different loss components, such as defocus, symmetry and photometric loss.

Li *et al.* [88] present occlusion pattern-aware network alongside occlusion pattern-aware loss. This loss helps to filter away the occluded views by verifying the photometric consistency. Network is designed on extracting features from sub-LFs in the similar manner to [138], followed by transformer network for the feature fusion. Regressed disparity in this method is refined by the warping-based aligning with central LF view.

Method of Zhang *et al.* [182] uses a combination of two networks, designed for disparity and occlusion prediction. First network uses multi-view feature matching to construct the coarse and residual matching cost, regression of which provides two disparity maps, which are further combined to the final disparity map. Second network assists the disparity estimation based on photometric consistency.

Since not many unsupervised estimation methods are available for the LF, it is worth considering similar methods from the field of stereo vision and optical flow in connection with the similarities to multi-view stereo discussed in Section 2.3.3. Based on the scene rigidity assumption, method of Wang *et al.* [161] estimates optical flow and stereo depth from videos in an unsupervised manner. Huang *et al.* [60] design an encoder-decoder network with mutual epipolar attention module for the self-supervised stereo depth estimation.

The first attempt to use the Census transform for deep learning was made by Meister *et al.* [100]. It is used as a replacement of brightness constancy constraint for the optical flow estimation. Details of their census implementation for the differentiable case, however, are not provided. In work of Juefei-Xu *et al.* [73] the modified local binary patterns, which are similar to census transform, are obtained by the convolutions with weighted sum of the intermediate images. Based on this, authors design modified neural network layers for the further processing of classification tasks.

3.3 Conclusion

Existing algorithms for calibrating LF cameras are quite efficient. However, not in all cases their implementation both on the software and hardware side is an easy task. Also, these algorithms do not cover the issue of changing the calibration parameters on-the-fly.

The main problem at the beginning of this dissertation was the lack of suitable SoA methods for running the algorithms under the constraints of the embedded hardware. In addition to the limitations in the availability of the hardware itself, the main problem was the extremely low computation time of the algorithm results. At the same time, analyzing such methods, most of which are in the field of deep learning, it can be noted that the hardware requirements necessary for their work are at a level higher than can provide embedded hardware, so at the moment it makes sense to return to the classical paradigm for the estimation of depth maps from LF.

The following chapters will describe how we dealt with these problems when designing our system for depth estimation from LF.

Light Field Calibration

This chapter covers the pipeline of LF camera calibration, which was developed and tested on real-world LF camera within the scope of the dissertation. It describes of LF calibration algorithm, LF views rectification parameters estimation, refinement of calibration data from arbitrary scene information and the findings on the LF auto-calibration. The presented algorithms were verified on a LF camera with single full-size lenses in an array, details of which will be provided in Chapter 6. It can be potentially used in micro-lens-based cameras, however, the potential misplacement of lenses in a manufactured array is not as large as in cameras with full-size lenses.

4.1 Light field calibration algorithm

The explanation of the used algorithm for LF camera array calibration starts with the basic concept of single camera calibration procedure.

4.1.1 Pattern-based camera calibration

As stated in Section 2.1.10, calibration is needed for estimation of camera intrinsic parameters, accuracy of which is vital for the proper 3D reconstruction. One of the most popular and at the same time easy to perform method of calibration is calibration using a pattern. The following will describe the calibration method proposed by Zhengyou Zhang in [184] in the way it was used for the LF camera calibration in this dissertation.

The single camera calibration technique proposed in [184] is the following:

1. Planar calibration pattern creation;
2. Capturing of the calibration pattern pictures by moving either the camera or the pattern;
3. Points of interest (*e.g.* corners) detection from these images;
4. Initial camera parameters estimation;
5. Radial distortion coefficients estimation;
6. Parameters refinement;

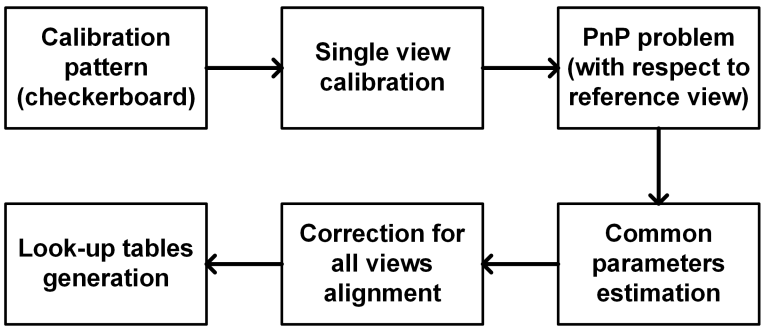


FIGURE 4.1: A pipeline of pattern-based calibration algorithm

Calibration pattern in this case is considered to be a planar structure with known geometry and with easily distinguishable feature points. The known geometry refers to the data on the relative location of the features in the calibration pattern space. It is important to be able to unambiguously obtain the relative location of the 3D points corresponding to the projections of these points on the image space, and accordingly to the pattern's features. Information about the position of these points will be used later at the stage of calculating the camera parameters.

The requirements for features from the calibration pattern are unambiguity of their location and ease of extraction from the general image space. Zhang suggested using a flat pattern with evenly spaced black squares on white, where the chips were the four corners of each square. However, more practical use is made of patterns based on the "chessboard"/"checkerboard" configuration, in which features are extracted from the corners where two white and two black squares intersect. Fig. 4.2 demonstrate how both pattern configurations look like.

There are several reasons for choosing the checkerboard-based pattern over the square-based. First, unlike the configuration with ordinary squares, in the case of the checkerboard it is easy to make a configuration that unambiguously reflects the orientation of the pattern in space. If you rotate the board with squares by 180° degrees, it is difficult to match the changed corners of the pattern with the original 3D points. At the same time, in the case of a checkerboard-based pattern with different parity of its sides, it is possible to uniquely determine the orientation of the pattern.

Another motivation for using the checkerboard-based pattern related to the subpixel refinement of the extracted features coordinates. When subpixel refinement is based on the corner's surroundings, the accuracy of its result is expected to be higher for the case of contrastive surroundings, which is obvious for the case of the checkerboard.

In our algorithm 2D corner points are extracted from images using the method proposed by Suzuki and Abe in [148]. Their positions are further refined to achieve subpixel accuracy by the method of Förstner and Gülch [36].

Without going into the details of how these methods work, since the details are of no interest within the scope of this dissertation, let us assume that for every LF view $k \in L_{:,:,s,t}, s \in S, t \in T$, and for every image with

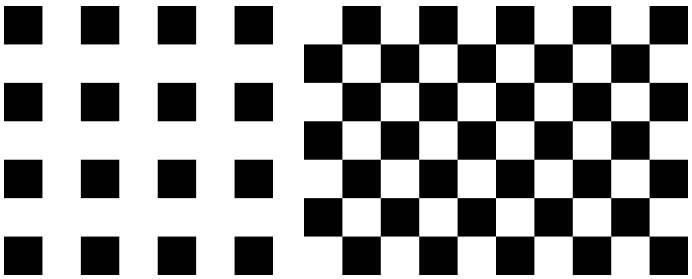


FIGURE 4.2: Examples of calibration patterns: (a) squares-based, (b) checkerboard-based.

pattern $i \in C$ captured on this view, where C denotes the totality of the captured images for a particular view k from the LF, we get a set of 2D points $\{m\}_{ki}$, which corresponds to detected pattern features. Based on the known geometry of the calibration pattern, a common 3D set of points $\{M\}$ can be obtained. Important for these points is that the third coordinate, which is responsible for the depth of the point, is always zero, and the other two coordinates are assigned by multiplying the length of the edge of the square at the base of the pattern by the relative position of the particular corner.

Using both $\{m\}_{k,}$ and $\{M\}$, the initial solution of matrix K from Eq. 2.6 can be obtained by applying direct linear transformation (DLT) [147]. To estimate distortion coefficients d, all the points for particular view are combined to the matrix according to the Eq. 2.15 and subjected to LLS [179]. Particular equations are provided in the original paper [184] and omitted here for simplicity.

Using the initial solution for K and d we can improve it by applying the Levenberg-Marquardt algorithm (LMA) [104] for nonlinear reduction of the reprojection error. For each pattern in each LF view, there is a unique rotation vector and translation vector that orient that pattern on the 3D scene. With the information about pattern placement and initial K, for every image we can build individual projection matrix P_{ki} from Eq. 2.12 and compute the set of reprojection of $\{M\}$ to image space as $\{\hat{m}\}_{ki}$ using Eq. 2.13. 2D points undistortion is applied following Eq. 2.15.

Resulting points per view k are used as a subject for minimization by LMA as:

$$\sum_{i=1}^{|C|} \|\{m\}_{ki} - \{\hat{m}\}_{ki}\| \tag{4.1}$$

where $\{\hat{m}\}_{ki}$ is retrieved by a reprojection function $Q()$ as:

$$Q(X, P, d) = PX \begin{bmatrix} r & r^2 \end{bmatrix} \begin{bmatrix} k1 \\ k2 \end{bmatrix}, \tag{4.2}$$

where r is obtained using Eq. 2.14, and $k1, k2 \in d$.

4.1.2 Calibration extension for multi-view cases

The previous considered equations describe the process of obtaining the parameters of the single LF view. If all the views observed the same calibration pattern, it is possible to do the estimations of LF views relative position by using the common set of 3D points $\{M\}$. Due to the planar (or close-to-planar in case of some real-world configurations) placement of the capturing sensors, we can assume that changes in the relative views placement are specified only by two coordinates. After that, to bring all views to similar image plane, the common camera parameters and ideal rotation and translation vectors can be estimated.

Using all observations of the same calibration pattern in all views C, in order to find view relative placement, we solve perspective-n-point (PnP) problem [35]. In general, the PnP problem can be solved with the above-mentioned DLT. Results of the estimation should be further refined by LMA.

After this step the set of rotation vectors $\{r_k^O\}$ and translation vectors $\{t_k^O\}$ per view are obtained. Rotation vectors are converted from rotations matrices by applying the Rodrigues transformation [129] to them.

Having the K_k matrices per view, the next step is to find a common intrinsic matrix K_r, which will be used for the rectification of all LF views to the common image space and is defined as the following:

$$\begin{gathered} f_r^x = \frac{\sum_{k=1}^{|C|} f_k^x}{|C|}; f_r^y = \frac{\sum_{k=1}^{|C|} f_k^y}{|C|}; \\ c_r^x = \frac{w}{2}; c_r^y = \frac{h}{2}; \\ K_r = \begin{bmatrix} f_r^x & 0 & c_r^x \\ 0 & f_r^y & c_r^y \\ 0 & 0 & 1 \end{bmatrix}, \end{gathered} \tag{4.3}$$

where C is the number of LF views, and w, h correspond to the width and height of each view, which are assumed to be the same.

Rotation and translation vectors r_k^O, t_k^O are used for getting an optimal value of common rotation and translation vector, to which all views would be brought after the rectification. Common rotation vector r_r is estimated as an average of all rotation vectors as:

$$r_r = \frac{\sum_{k=1}^{|C|} r_k^O}{|C| - 1}, \tag{4.4}$$

and further converted to rotation matrix R_r by applying Rodrigues transform.

Computations of common translation vector involve the spatial information of every view. Since the translation vectors are defined *w.r.t.* reference view, we estimate the relative values by involving S and T as vertical and horizontal spatial dimensions of LF, where $S \times T = |C|$.

For a specific LF view k, the translation vector t_i^V is formed from per-axis components as $t_k^V = [t_i^{V_x}\ t_i^{V_y}\ t_i^{V_z}]$ and estimated as a mean of all components not in the same row or column as:

$$\begin{gathered} a = \lfloor k/S \rfloor; b = k \mod S; \\ t_k^{V_x} = \begin{cases} t_k^{O_x}/(\hat{b}-b), \hat{b}-b \neq 0 \\ 0, \hat{b}-b = 0 \end{cases} \\ t_k^{V_y} = \begin{cases} t_k^{O_y}/(\hat{a}-a), \hat{a}-a \neq 0 \\ 0, \hat{a}-a = 0 \end{cases} \\ t_k^{V_z} = 0 \end{gathered}, \tag{4.5}$$

where a, b stand for spatial coordinates of any LF view, and $\hat{a}, \hat{b}$ stand for spatial coordinates of reference view.

All translation vectors form a set t^V. Common translation vector $\bar{t_r} = [t_r^x\ t_r^y\ 0]$ is estimated as:

$$\begin{gathered} S_{t^x} = \sum_{k=1}^{|C|}(1-\delta_{t_k^{V_x},0}); S_{t^y} = \sum_{k=1}^{|C|}(1-\delta_{t_k^{V_y},0}) \\ t_r^x = \frac{\sum_{k=1}^{|C|} t_k^{V_x}}{S_{t^x}}; t_r^y = \frac{\sum_{k=1}^{|C|} t_k^{V_y}}{S_{t^y}} \end{gathered}, \tag{4.6}$$

where S_{t_x} and S_{t_y} are numbers of particular non-zero components in t^V, found using Kronecker delta:

$$\delta_{x,y} = \begin{cases} 0, x \neq y \\ 1, x = y \end{cases}. \tag{4.7}$$

This vector is a basis for the set of per-view translation vectors t^p:

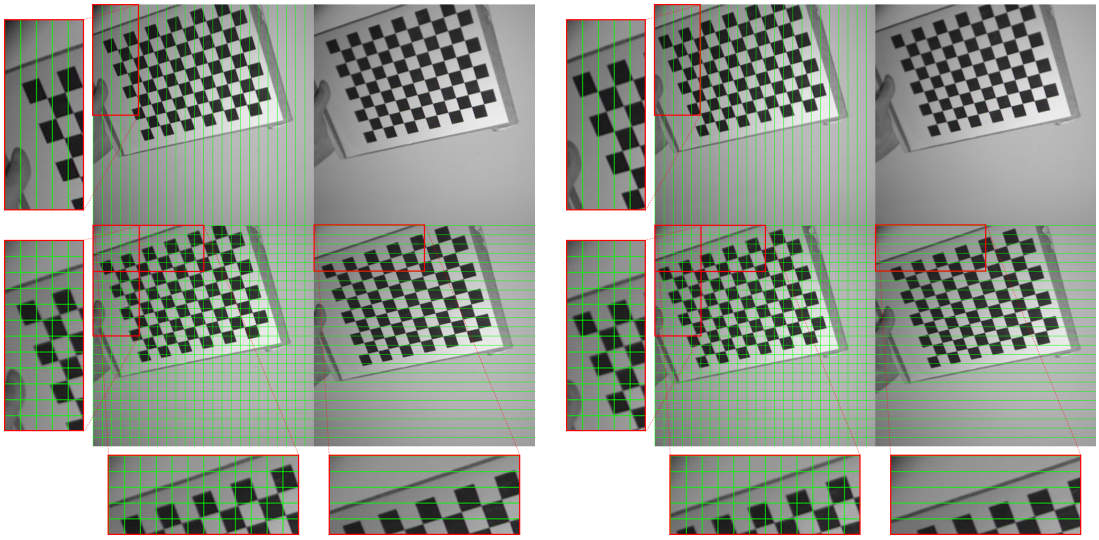
$$t_k^{p_x} = t_r^x(\hat{b}-\bar{b}); t_k^{p_y} = t_r^y(\hat{a}-\bar{a}); t_k^{p_z} = 0. \tag{4.8}$$

Using the results from Eq. 4.3–4.8 we estimate the rectified projection matrix P_r per every LF view as:

$$\begin{gathered} \bar{R_k} = R_r R_k^{O^T} \\ \bar{t_k} = \bar{R_k} t_k^p \\ P_k^r = K_r[\bar{R_k}|\bar{t_k}] \end{gathered} \tag{4.9}$$

The resulting camera projection matrix represents the ideal position of its corresponding LF view.

The further remapping of pixels to the proper position is done by applying a LUT, which stores per-pixel coordinates of the rectified image and generated using P_k^r, distortion coefficients and original intrinsic and extrinsic values. Fig. 4.3 shows, how LF images changes after applying the described



(a) (b)

FIGURE 4.3: A subset of (a) original and (b) rectified LF after applying the presented algorithm.

pipeline.

4.2 Calibration Auto-Refinement

An auto-refinement algorithm uses the previously found reference calibration and tries to estimate, how the calibration parameters need to be compensated for the current configuration of the multi-view camera. It is needed in cases when the placement of views was changed during the camera exploitation, *e.g.* camera is mounted on the car and it is a subject of shakes and other mechanical influences. A pipeline of this method is presented in Fig. 4.4.

The important criteria for the selection of correspondences detection algorithm were the robustness of features in real-world images and the running time of the algorithm. Among existing methods, a combination of "good features to track (GFTT)" method for features extraction [137] with KLT feature tracker [152] was chosen. Both of these algorithms are considered as computationally efficient due to the simplicity of the underlying operations, which was our main criterion for choosing this combination of algorithms.

The GFTT algorithm identifies feature points in the image, which are likely to be tracked consistently in the consecutive frames. Usually good candidates for the accurate features are the corners of the objects. The algorithm works by calculating corner response function for each pixel in the image, which measures the intensity changes in the neighborhood of the pixel.

KLT uses spatial intensity gradients of the image to find the positions that are most likely to correspond to the best match. This approach is faster than traditional methods because it involves examination of a much smaller set of potential image matches. In addition, the algorithm can be adapted to handle image rotation, zooming, and shifting.

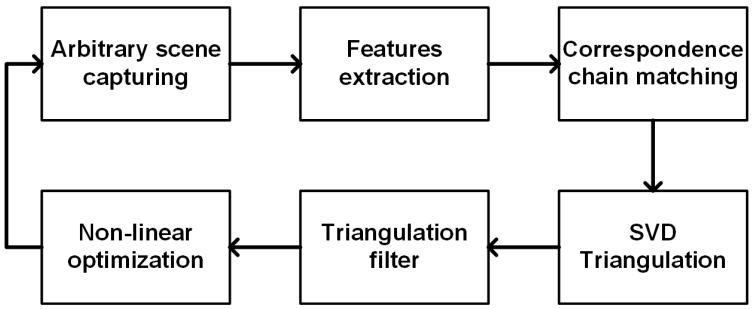


FIGURE 4.4: A pipeline of auto-refinement algorithm

Exploiting the multi-view nature of the LF, in particular the fact that the projection of 3D points from the scene can be seen in all views, the determined features are tracked between different views in a chain manner. Correspondences from the reference view are verified in the neighboring view on the same axis, tracked ones are searched in the next view, and on the last view in one row features are matched in the upper one. This principle is demonstrated in Fig. 4.5.

Such a method is needed for reducing the number of possibly mismatched correspondences, which may occur especially in the small-baseline LF systems, like the camera used in Chapter 4.4. This method also helps for the verification of the correspondence inaccuracies, since the very strong matches have to be preserved in all views.

Additional filtration, based on the estimation of fundamental matrix between correspondences in adjacent views, and exclusion of non-matched features is applied. Having a fundamental matrix F, estimated by *e.g.* random sample consensus (RANSAC) method [35], the points from neighboring view x_1 and x_2 are checked by the value of $x_2^T F x_1$ being lower than a certain threshold.

There's a small number of features, which were extracted more than once from the image. To remove the possible influence of such correspondences, we preliminary check the result of the feature detection algorithm by searching of the nearby correspondences using the Euclidean distance between points.

By empirically setting a certain threshold for these distances we can efficiently filter out the closely placed features. For our experiments it was set to $\sqrt{2}$.

Previously described methods of filtration can eliminate a big amount of wrongly estimated correspondences. However, some false matches, especially the ones placed close or on the texture-less areas, can survive these checks. To eliminate such mismatches we propose a filter, based on the reprojection of triangulated points.

The filtered points are triangulated based on original intrinsic values K_k and rotation and translation vectors R_k^O, t_k^O. A projection matrix P_k^O is composed as $K_k[R_k^O|t_k^O]$. For every correspondence $m_k = [x_k, y_k]$, matched in every view out of C, by taking a vector P_{ok}^{rT} for every row of the corresponding

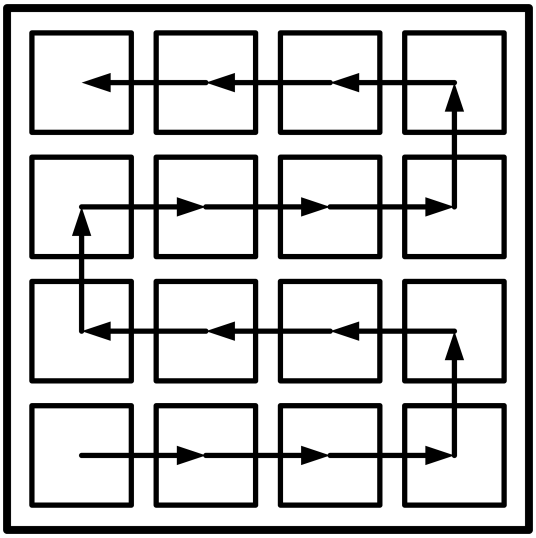


FIGURE 4.5: Visualization of views traversing in chain manner

projection matrix the matrix A is composed as follows [46]:

$$A = \begin{bmatrix} x_1 P_{O1}^{3T} - P_{O1}^{1T} \\ y_1 P_{O1}^{3T} - P_{O1}^{2T} \\ \vdots \\ x_N P_{ON}^{3T} - P_{ON}^{1T} \\ y_N P_{ON}^{3T} - P_{ON}^{2T} \end{bmatrix} \tag{4.10}$$

singular value decomposition (SVD) is applied to this matrix, the triangulated points are extracted from the smallest singular value of A.

Using the P_O we project the triangulated points M_t to 2D space and estimate the Euclidean distance between original and projected points:

$$\begin{aligned} m_t &= PM_t \\ dist_t &= \|m_t - m_r\|, \end{aligned} \tag{4.11}$$

where m_r stands for an original correspondence from reference view.

We find a median of all distances between each correspondence and its projection, and filter out all matches, for which the distance is bigger than the median value.

To finally estimate the the compensation for intrinsic and extrinsic values, a minimization problem involving criterion from Eq. 4.1 is solved. In this case, since the 3D points coordinates are also involved, it is wrapped as bundle adjustment [154].

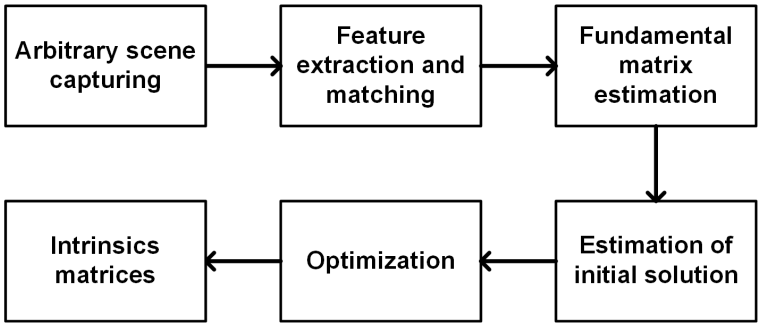


FIGURE 4.6: A pipeline of intrinsics auto-calibration algorithm

4.3 Findings on Light Field Auto-Calibration

This section is covering the work, done within this dissertation towards solving full autocalibration problem [1].

Similar to auto-refinement, auto-calibration relies on the data from arbitrary scene with some constraints. However, there is no preliminary information about the camera parameters, which makes the calibration harder. One of the methods to do the auto-calibration is to form some kind of an absolute target. The most frequently used example of such a target is an absolute conic [46].

Considering a plane at infinity π_∞, absolute conic is an imaginary conic on this plane, points on which satisfy:

$$X_1^2 + X_2^2 + X_3^2 = X4 = 0. \tag{4.12}$$

Analysis of absolute conic properties gives the information about camera intrinsic parameters.

Fig. 4.6 shows the workflow of the developed LF autocalibration method. The method proposed by Lourakis in [95] was chosen as the basis.

The method is based on the computation of fundamental matrices from the scene images pairs. Estimation of the matrices is done on top of the matched features. Similar to auto-refinement case, experiments were done on the combination of GFTT with KLT. Together with that, a method called Scale-Invariant Feature Transform (SIFT) [96] was tested.

Pipeline of the SIFT algorithm can be described as follows. First, potential feature points are found using a scale-space extrema detection algorithm. It is done by applying the convolution with Gaussian filters at the difference scales to the image and finding the difference between the resulting images, forming the difference of Gaussians (DoG). To reduce the number of feature candidates, the proper keypoints are localized by comparing the DoG images. Based on the finite differences around the survived candidates and the resulting dominant gradient direction, orientation is assigned to the feature. Descriptor of the keypoint is estimated by finding the local gradient-based features and constructing the histogram of gradient orientation. Further,

[1] This section is partially based on the master thesis "Auto-Calibration Algorithms for Light Fields", written by Ngoc Thy My Nguyen under our supervision

matches of different keypoints within different images can be found based on these descriptors.

The matched correspondences are further used to find the so-called fundamental matrix between two views. Fundamental matrix F [46] described the relation of corresponding points in two views. It is a 3x3 matrix, which keeps the geometrical relationships between the views and allows the estimation of epipolar geometry between these views. It can be formed as:

$$F = K_1^{-T}[t]_\times R K_2^{-1}, \tag{4.13}$$

where K_1 and K_2 are the intrinsic matrices of two cameras, and $[t]_\times R$ described the relative placement of these cameras, where for $t = (t_1, t_2, t_3)$:

$$\begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}. \tag{4.14}$$

In the ideal case, having homogeneous coordinates of the image points x_1 and x_2 from these cameras, their relations can be described by:

$$x_1^T F x_2 = 0 \tag{4.15}$$

This equation describes the fact that the epipolar lines in one view are constrained to pass through the corresponding point in the other view.

There are different methods for the fundamental matrix estimation. The straightforward approach involves the eight correnspondences, forming a system of equations, which can be solved by finding a non-trivial solution of $Ax = 0$ by the least squares [46]. This method usually implies points normalization for improving its accuracy. While being the computationally efficient, this method is very sensitive to outliers and noise, potentially providing unstable and even unusable solutions.

A good alternative to the direct fundamental matrix estimation is its RANSAC-based extension. By this method a random minimal set of correspondences is used to for the eight-point fundamental matrix estimation, and the accuracy of the estimation is verified on all the point correspondences by finding the fundamental matrix which produces the maximum number of inliers, or the points for which the relationship described by Eq. 4.15 holds to a certain threshold T_F as:

$$x_1^T F x_2 < T_F \tag{4.16}$$

For the additional accuracy increase, a gold standard (GS) can be used. It minimizes the reprojection error between the matched points in the two views in an iterative manner. Having initial fundamental matrix, it is refined by minimizing the sum of squared distances between the observed and predicted image points. Besides being the computationally intensive algorithm, another disadvantage of this method is the necessity of Gaussian distribution of the image noises.

Obtained pairwise fundamental matrices form a core for their further processing by building Kruppa equations [34] on them. They used for the estimation of camera intrinsic parameters. Below a derivation of the simpified Kruppa equations will be shown.

An essential matrix E is defined as $E = [t]_\times R$. Based on this and using the Eq. 4.13, F can be redefined as:

$$F = K^{-T} E K^{-1}, \tag{4.17}$$

out of which E is expressed as:

$$E = K^T F K^1. \tag{4.18}$$

Work of Triverdi [155] states that the symmetric matrix EE^T is independent from the rotation matrix R, since

$$EE^T = [t]_\times R R^T ([t]_\times)^T = [t]_\times ([t]_\times)^T. \tag{4.19}$$

For the further estimations we denote a symmetric matrix A as:

$$A = KK^T = \begin{bmatrix} f_x^2 + c_x^2 & c_x c_y & c_x \\ c_x c_y & f_y^2 + c_y^2 & c_y \\ c_x & c_y & 1 \end{bmatrix}, \tag{4.20}$$

based on which by substitution we obtain:

$$FAF^T = K^{-T} [t]_\times ([t]_\times)^T K^{-1} \tag{4.21}$$

Epipoles e_1 and e_2 defined as a point in one image that corresponds to the position of the camera center in the other image. In other words, it can be determined as the point of intersection between the line of sight of a camera center and the image plane of the other camera. Using the epipolar geometry, we can state that $F^T e_2 = 0$. By Eq. 4.13 e_2 must satisfy the following:

$$F = K^{-T} R^T ([t]_\times)^T K^{-1} e_2 = 0, \tag{4.22}$$

from which e_2 can be found as:

$$e_2 = \lambda K t, \tag{4.23}$$

where λ is a non-zero scalar. From 4.23 t can be expressed as:

$$t = \frac{K^{-1} e_2}{\lambda}, \tag{4.24}$$

and the original Kruppa equations are then formed as:

$$FAF^T = \gamma [e_2]_\times A ([e_2]_\times)^T \tag{4.25}$$

Simplified Kruppa equations are considered to be more practically useful for the tasks of auto-calibration, since they do not involve the estimation of epipole, which is especially important for the real camera cases with the noise presence. Simplified equations are based on the factorization of the fundamental matrix based on SVD [39]. It decomposes a matrix into three matrices: a left singular matrix U, a diagonal matrix of singular values D, and a right singular matrix V^T:

$$SVD(F) = UDV^T. \tag{4.26}$$

Matrix D is having the form:

$$\begin{bmatrix} r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.27}$$

where r and s are eigenvalues of the matrix FF^T.

In this case, the epipole e_2 can be deduced and the skew matrix $[e_2]_\times$ is found as:

$$F^T e_2 = VD^T U^T e_2 = 0 \tag{4.28}$$

$$e_2 = \delta U m, \delta \neq 0, m = [0,0,1]^T \tag{4.29}$$

$$[e_2]_\times = \mu U M U^T, \tag{4.30}$$

where μ is a nonzero scale factor and $M = [m]_\times$ is of the form:

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.31}$$

Based on the listed equations, the new group of Kruppa equations can be defined as:

$$FAF^T = \mu U M U^T A U M^T U^T, \tag{4.32}$$

out of which the simplified Kruppa equations are obtained based on multiplication of left and right equations parts by orthogonal U and U^T:

$$DV^T AVD^T = \mu M U^T A U M^T. \tag{4.33}$$

Eq. 4.33 provides three linearly dependent equations. If u_1, u_2, u_3 are the column vectors of U, and v_1, v_2, v_3 are the column vectors of V, the Eq. 4.33 can be expressed as:

$$DV^T AVD^T = \begin{bmatrix} r^2 v_1^T A v_1 & rs v_1^T A v_2 & 0 \\ rs v_2^T A v_1 & s^2 v_2^T A v_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.34}$$

$$MU^T AUM^T = \begin{bmatrix} u_2^T A u_2 & -u_2^T A u_1 & 0 \\ -u_1^T A u_2 & u_1^T A u_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.35}$$

Out of it, three equations can be derived as:

$$\frac{r^2 v_1^T A v_1}{u_2^T A u_2} = \frac{rs v_1^T A v_2}{-u_2^T A u_1} = \frac{s^2 v_2^T A v_2}{u_1^T A u_1} \tag{4.36}$$

The simplified equations allow to determine an intrinsic internal matrix, which serves as a starting point for the optimization process.

If $S_F = [r, s, u_1^T, u_2^T, u_3^T, v_1^T, v_2^T, v_3^T]$ is a vector of SVD components of the fundamental matrix, then by using three ration of Eq. 4.36:

$$\frac{\rho_i(S_F, A)}{\phi_i(S_F, A)} \tag{4.37}$$

we get two polynomial equations per fundamental matrix as:

$$\rho_1(S_F, A)\phi_2(S_F, A) - \rho_2(S_F, A)\phi_1(S_F, A) = 0 \tag{4.38}$$
$$\rho_1(S_F, A)\phi_3(S_F, A) - \rho_3(S_F, A)\phi_1(S_F, A) = 0 \tag{4.39}$$

These equations can be treated as quadratic and solved separately. Assuming that aspect ratio for our case is close to one, we can state that $A_{11} = A_{22}$, hence $f_x = f_y$. For N images we get $(N-1)$ fundamental matrices, and each of them may have up to 4 solutions. However, not all of these solutions are valid since they can make the matrix A either unreal or not positive definite.

Quadratic equations have an advantage because they do not require an initial guess, and solutions can be easily found by considering one equation at a time. However, this can lead to solutions that are far from the correct values. Therefore, we need to select appropriate initial solutions for the optimization steps. Some strategies can be used to choose which initial solutions to use, so that only meaningful solutions are considered.

An alternative to the quadratic equations is the Newton-Raphson method. The Newton-Raphson method for a nonlinear system of equations considers all equations of the system simultaneously, unlike the quadratic equations method. However, this method requires a starting point to initiate the iteration. If the initial guess is close to the actual solution, then the method is very efficient and can ensure convergence to a root.

To estimate a good starting point for the optimization process, in the original publication Lourakis suggests several strategies, including choosing a solution randomly or using the average value of all solutions. In our work, we have chosen to use the second option, where we compute the average value of all valid solutions and pass this value to the optimization process. Having an initial solution, two approaches for the refinement are considered, namely LMA and particle swarm optimization (PSO) [19].

For conducting the optimization via LMA, we construct the residual equation based on Lourakis *et al.* [95]. Difference of ratios between equations in

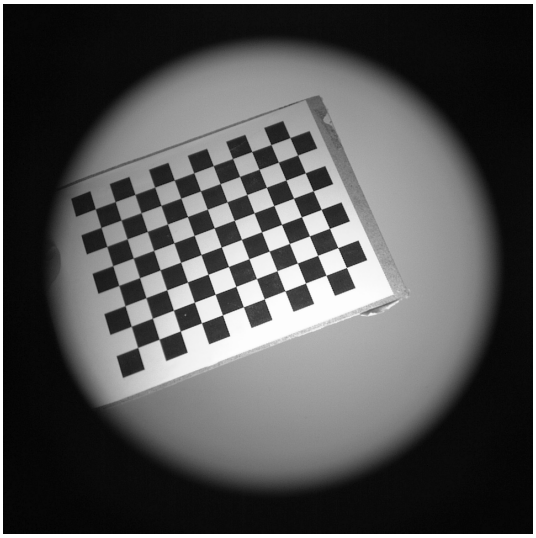


FIGURE 4.7: An example of checkerboard pattern, used for LF camera calibration

Eq. 4.39 is denoted as:

$$\pi_{ij}(S_F, A) = \frac{\rho_i(S_F, A)}{\phi_i(S_F, A)} - \frac{\rho_j(S_F, A)}{\phi_j(S_F, A)}, \tag{4.40}$$

where $i, j = 1..3$. Variance of π_{ij} can be approximated as:

$$\sigma^2_{\pi_{ij}}(S_F, A) = \frac{\partial \pi_{ij}(S_F, A)}{\partial S_F} \Lambda_{S_F} \frac{\partial \pi_{ij}(S_F, A)^T}{\partial S_F}, \tag{4.41}$$

where $\frac{\partial \pi_{ij}(S_F, A)}{\partial S_F}$ is a 1×20 derivative vector of $\pi_{ij}(S_F, A)$ at S_F and Λ_{S_F} is a 20×20 covariance matrix, associated with S_F. This covariance matrix can be approximated as:

$$\Lambda_{S_F} = \frac{\partial S_F}{\partial F} \Lambda_F \frac{\partial S_F}{\partial F}^T, \tag{4.42}$$

where Λ_F is a 9×9 covariance matrix of the fundamental matrix F, computed using the method from Csurka *et al.* [30], and $\frac{\partial S_F}{\partial F}$ is a 20×9 matrix containing the value of the Jacobian matrix of S_F at F, which can be found using the method of Papadopoulo and Lourakis [118].

Matrix A is then optimized by LMA as:

$$A = argmin \sum_{i=1}^{N-1} \frac{\pi^2_{12}(S_{Fi}, A)}{\sigma^2_{\pi_{12}}(S_{Fi}, A)} + \frac{\pi^2_{13}(S_{Fi}, A)}{\sigma^2_{\pi_{13}}(S_{Fi}, A)} + \frac{\pi^2_{23}(S_{Fi}, A)}{\sigma^2_{\pi_{23}}(S_{Fi}, A)}. \tag{4.43}$$

A has five unknowns, so the minimum number of input images needed for the optimization process to be successful is more than three. In general, the more input images available, the more constraints are placed on the optimization process, which provides more accurate solution.

After optimizing the A, recalling that $A = KK^T$, we apply the Cholesky

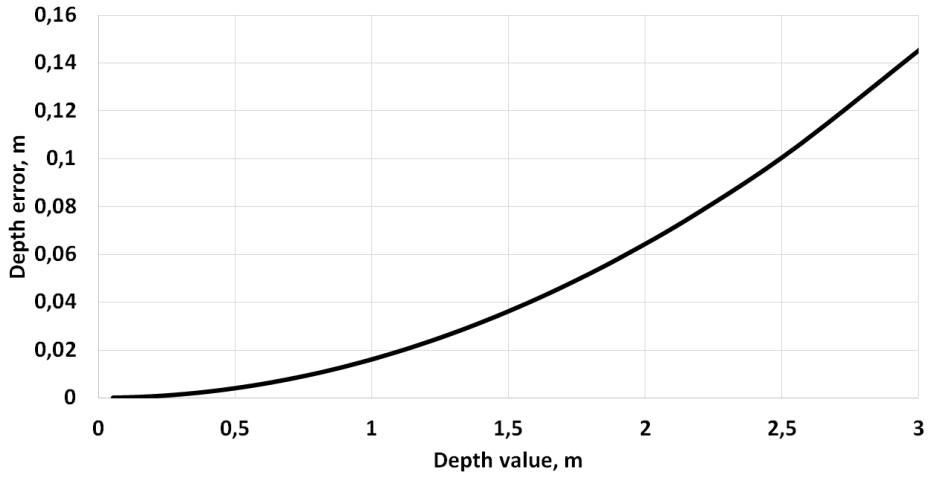


FIGURE 4.8: Depth error for $E_{PnP} = 0.246$ pixels

decomposition [12] to obtain K^{-T}, which after transposing and inversion serves as intrinsic matrix.

4.4 Experiments

4.4.1 Calibration algorithm

For the tests of pattern-based calibration algorithm we used a custom LF camera, described in Chapter 6. It composed of 4×4 lenses, providing a 960×960 pixels RGB image per view, which afterward converted to grayscale for the sake of calculations simplicity. A 12×9 checkerboard pattern with 20×20 mm squares was used for the calibration. An example of the calibration pattern is presented on Fig. 4.7. We captured 40 scenes with this pattern for testing purposes.

To control the accuracy of the estimated intrinsic matrix K and distortion coefficients d, the reprojection error is computed as in Eq. 4.1 for every LF view. Similar computations are done during the estimation of the extrinsic values.

For the test dataset the average reprojection errors for monocular calibration and PnP problem were $E_{mono} = 0.159$ and $E_{PnP} = 0.246$ pixels respectively.

A comparison of our method was done with the method of Xu *et al.* [172]. For the same test dataset we have obtained reprojection errors of $E_{mono} = 0.153$ and $E_{PnP} = 0.156$ pixels.

We can state that optimization of all parameters together for the precisely estimated points make sense in terms of accuracy. However, the potential drawback is related to higher running time of the joined optimization.

To verify influence of the reprojection error to the actual depth reconstruction we found a depth error with common focal length values $f = f_x = f_y = 850$ pix and baseline between the two most distant views on the same axis $b = 0.018$ m as:

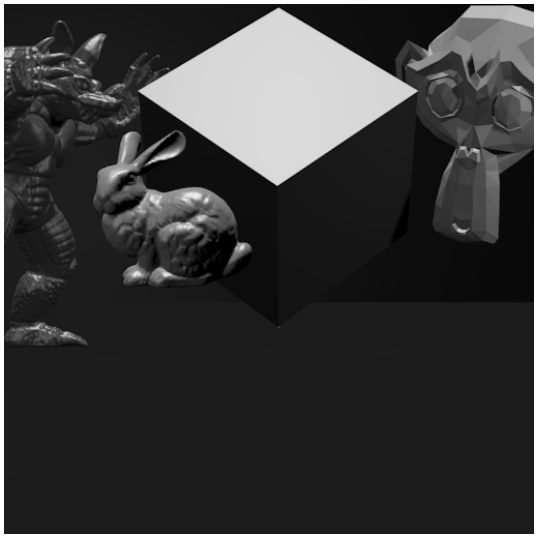


FIGURE 4.9: An example of synthetic scene for verifying the auto-refinement algorithm

$$\Delta Z = \frac{fb}{d - E_{pnp}} + \frac{fb}{d + E_{pnp}}, \tag{4.44}$$

where d is the disparity value, *i.e.* displacement between pixels, which can be further converted to actual depth value. This error is visualized on Fig. 4.8.

On a target range of the test camera, which is 0.5-2.0 m, the estimated reprojection introduces an inaccuracy in the amount of 0.004-0.065 m, which is lower then the depth accuracy on this specific distance range.

4.4.2 Auto-refinement algorithm

A dataset of synthetic LFs was used for verification of auto-refinement stability. The rectified and undistorted images with known intrinsic parameters were generated using Blender and the script from 4-dimensional LF Benchmark [1, 55]. 5×5 LFs of size 512×512 pixels per view with a baseline of 100 mm between adjacent views were generated from a simple scene with different overlapping objects, as demonstrated in Fig. 4.9. A sequence of ten images was used for the tests.

In average, 1/10 of originally detected points were filtered by correspondence chain matching, out of which half of the points survived the triangulation-based filtering. Processing of one frame takes around one second.

The difference between original and refined intrinsic and extrinsic parameters was measured and considered as non-informative, since no significant difference between original and refined values was found. It can be explained by the quality of correspondences, which is in general good for the synthetic data.

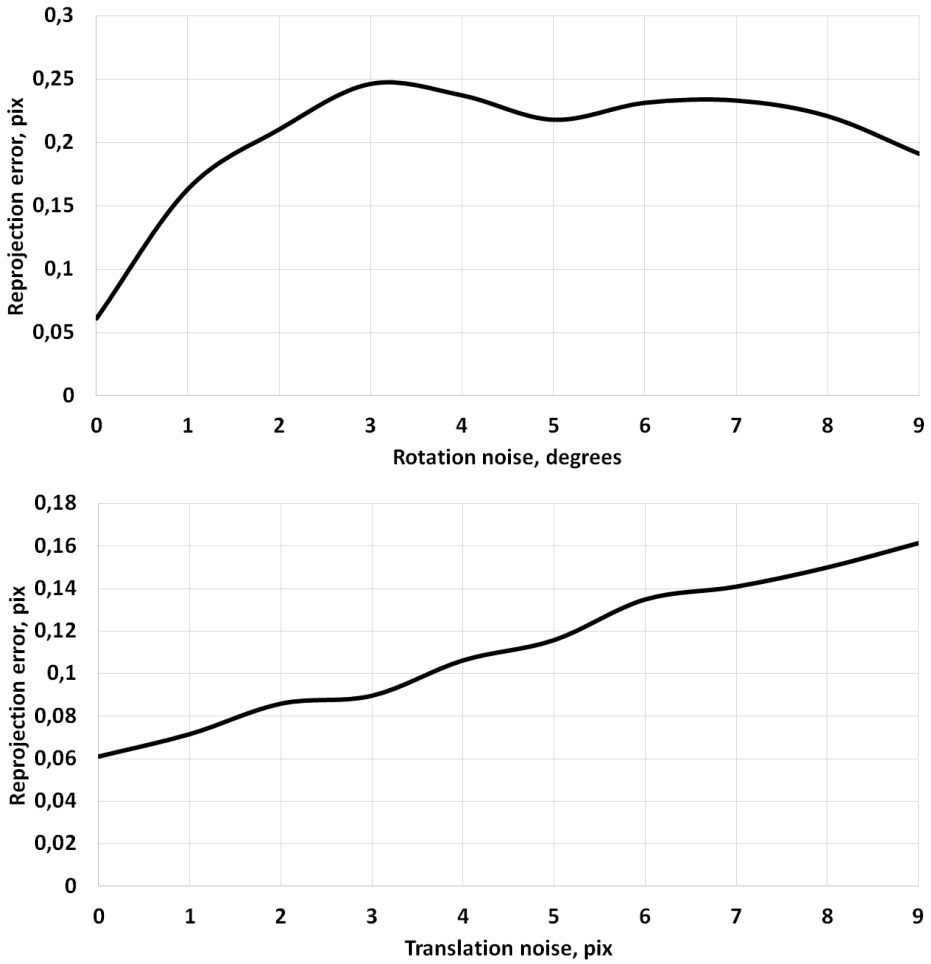


FIGURE 4.10: Reprojection error of auto-refinement algorithm dependently of the applied noise for synthetic images

To check the accuracy of the auto-refinement algorithm, we applied rotation and translation noise to the captured frames and measured the average reprojection errors. Results are demonstrated on Fig. 4.10.

In total both types of noise create acceptable level of reprojection error. In a similar to pattern-based calibration manner we have evaluated the depth error for maximum reprojection error from the rotation noise, result of which is presented on Fig. 4.11.

For the noise simulation of rotation was applied only to Z axis. Rotations on X and Y axes are the subject of tangential distortion. It occurs when the lens array is not parallel to the camera sensor plane. This type of distortion is assumed to be zero in the applied camera model. The pattern-based calibration method used gives adequate results in terms of the resulting reprojection error. Empirically, we have found that the overall calibration quality depends largely on the quality of the calibration target, especially its flatness.

For the proper calibration we have come to the number of 25-30 pattern images in one sequence. All areas on the LF views should be covered with pattern images in various positions to ensure correct estimation of internal values.

	f_x	f_y	c_x	c_y	f_x	f_y	c_x	c_y
Cam 1	881.91	884.22	483.58	444.74	875.23	878.01	495.24	380.42
Cam 2	883.25	885.25	481.79	450.06	874.69	879.53	481.08	455.63
Cam 3	878.58	879.69	472.57	451.03	876.54	878.08	554.69	360.00
Cam 4	875.47	875.05	462.79	449.23	877.99	877.89	509.58	558.97
Cam 5	878.86	882.33	486.36	453.20	877.91	877.98	484.64	473.27
Cam 6	877.71	879.89	477.94	454.04	877.04	879.40	499.58	397.84
Cam 7	875.80	877.29	473.75	453.78	878.52	878.51	477.51	471.94
Cam 8	880.45	881.08	469.48	461.32	878.52	878.51	447.21	485.04
Cam 9	891.05	895.21	488.83	459.81	878.32	878.24	489.16	492.78
Cam 10	873.15	875.59	483.40	461.65	877.53	878.26	493.65	510.49
Cam 11	879.39	881.20	477.26	465.99	877.53	878.26	493.65	510.49
Cam 12	879.62	880.37	471.45	467.93	877.24	878.37	426.24	527.44
Cam 13	871.50	874.76	488.55	466.21	877.43	878.37	434.25	360.00
Cam 14	878.40	881.41	485.91	466.95	878.18	878.11	462.24	389.90
Cam 15	872.14	874.12	477.87	467.60	877.95	878.21	472.86	489.73
Cam 16	875.24	876.00	471.87	474.72	877.91	878.20	486.59	458.45
Average	878.28	880.22	478.34	459.27	877.41	878.37	481.76	457.65
	(a)				(b)			

TABLE 4.1: Results of: (a) pattern-based calibration, (b) auto-calibration

One of the assumptions of the algorithms was the similarity of the lens parameters for each view. For cases with a significant shift (in terms of rotation or translation) of at least one of the views over the others, averaging over extrinsic values cannot be used; it should be replaced by nonlinear methods.

Experiments with the auto-refinement algorithm on synthetic images show that the reprojection error increases in proportion to the level of lens shift, while changes in their rotation affect only up to some point with a plateau thereafter.

During the auto-refinement experiments, we noticed that optimizing all the parameters together leads to incorrect results. This was the motivation for dividing the optimization procedure into three steps applied to the same model. First, only 3D points are optimized, while intrinsic and extrinsic camera values are fixed. This condition is relaxed in the second part, where the intrinsic values of all views are optimized. Finally, we optimize all camera parameters together. All of the above steps are repeated with new captured scenes. It can be stopped either by using a certain number of iterations or by reaching the desired reprojection error below a threshold value.

Several limitations of the auto-refinement algorithm were identified during the tests. It does not work well with repeating textures and with small

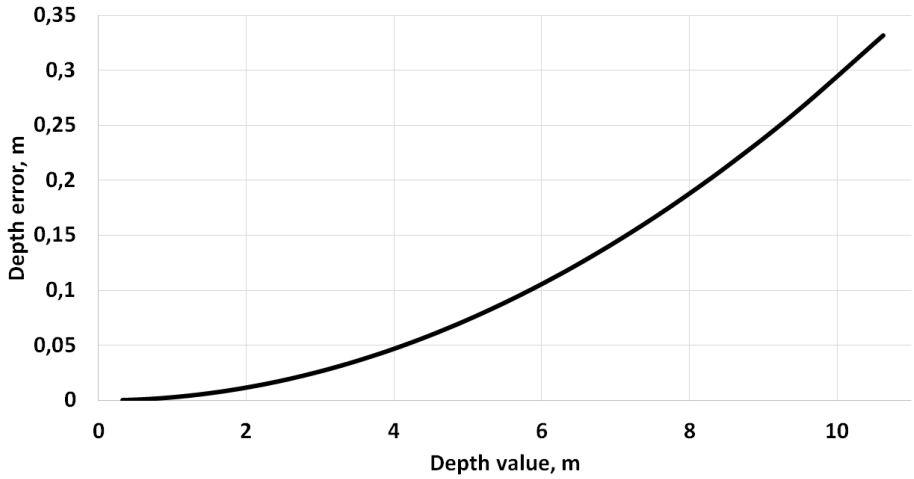


FIGURE 4.11: Auto-refinement depth error

distances between detected matches, *e.g.* on keyboards. This problem can be solved either by applying additional filtering measures or by changing the match detection algorithm. In addition, the coverage of the image area by the correspondence is important to obtain correct lens geometry, a similar requirement stands for pattern-based calibration. Additional correspondence distribution checks can be made to discard images without proper coverage. Because of this, the distortion coefficients cannot be corrected by our auto-rectification method and remain locked during the optimization stage.

We have tested the auto-refinement algorithm with and without triangulation filter. Without filtering the results were totally incorrect, so they are not included in part of the experiments.

4.4.3 Auto-calibration algorithm

Experiments for auto-calibration were done on the similar images set from 4×4 LF camera, presented in Section 6. Results of comparison of auto-calibration results with pattern-based calibration are presented in Tables 4.1 and 4.2.

SIFT was applied to arbitrary scene images for the features extraction. For each camera, matching points were determined by comparing the features of one image with those in the subsequent image, following the previously described correspondence chain matching. The surviving correspondences were then utilized to estimate the fundamental matrix using the normalized 8-point algorithm. While these matrices were sufficient for the next step, we applied the GS algorithm to enhance the accuracy of the fundamental results, which was the most time-consuming process in the entire procedure. After optimizing the fundamental matrices, we used the method presented in Section 4.3 to estimate the intrinsic matrices.

Estimated focal lengths were quite similar to those obtained using the pattern-based method. Unfortunately the method failed to properly estimate the camera centers, which are deviating in the relatively big range. Also, the intrinsic parameters obtained using this technique were unstable since

	f_x	f_y	c_x	c_y
Cam 1	6.68	6.21	11.66	64.32
Cam 2	8.56	5.72	0.71	5.57
Cam 3	2.04	1.61	82.12	91.03
Cam 4	2.52	2.84	46.79	109.74
Cam 5	0.95	4.35	1.72	20.07
Cam 6	0.67	0.49	21.64	56.20
Cam 7	2.72	1.22	3.76	18.16
Cam 8	1.93	2.57	22.27	23.72
Cam 9	12.73	16.97	0.33	32.97
Cam 10	4.38	2.67	10.25	48.84
Cam 11	1.86	2.94	16.39	44.50
Cam 12	2.38	2.00	45.21	59.51
Cam 13	5.93	3.61	54.30	106.21
Cam 14	0.22	3.30	23.67	77.05
Cam 15	5.81	4.09	5.01	22.13
Cam 16	2.67	2.20	14.72	16.27
Average	3.88	3.92	22.53	49.77

TABLE 4.2: Difference between Tables 4.1a and 4.1b

different image sequences of the same camera resulted in different intrinsic matrices. Since the primary process of estimating the internal camera parameters relied entirely on the SVD of the fundamental matrix, a small change in the values of fundamental matrices affected the final results of the estimation.

4.5 Conclusion

The presented algorithm for LF camera calibration is simple to implement because of the easy availability of the components needed to make it work. Also, it does not require high skills to calibrate with it, which makes it possible, for example, to delegate the task of calibrating LF cameras with it to personnel without additional training. The algorithm we have presented to compensate for changes in camera parameters allows us to potentially simplify their use in applications where recalibration is not feasible. The findings related to auto-calibration of LF cameras can be used in the future for the development of fully automatic calibration algorithm.

Geometrical depth estimation

This chapter covers the main algorithm of the dissertation – the actual depth estimation from LFs.

5.1 Algorithm outline

In general, algorithms for computing depth maps based on multiple 2D images have the following structure:

1. Correct the lens distortion and bring all images to the common intrinsic parameters (*i.e.* undistort and rectify images)
2. Measure the similarities between images
3. Construct a matching cost based on these similarities
4. Employ the optimization methods on the matching cost
5. Regress the disparity map from the optimized matching cost
6. Apply post-processing operations to the disparity map
7. Convert the post-processed disparity map to depth map

The first step for the presented algorithm is solved using the calibration method, described in the previous chapters. Further steps with respect to our algorithm will be explained in the following sections step by step.

5.1.1 Image similarity measurements

Estimation of disparity map from 2D images relies on the concept of similarity comparison between these images. By that we want to estimate what is the degree of coincidence of different image parts between each other. We will consider various methods for doing so.

Pixel methods

The first group of methods compares just the different pixels in images. Two most common functions for measuring the similarity of two pixels $p_1, p_2 \in R^d$ are [26]:

- $L1$, or Manhattan distance, defined as

$$\|p_1 - p_2\|_1 = \sum_{i=1}^{d} |p_{1_i} - p_{2_i}| \tag{5.1}$$

- $L2$, or Euclidean distance, defined as:

$$\|p_1 - p_2\|_2 = \sqrt{\sum_{i=1}^{d} (p_{1_i} - p_{2_i})^2} \tag{5.2}$$

While $L1$ comparison is faster to estimate, since it involves simple arithmetical operations, it might not provide enough accuracy due to the limited outliers sensitivity compared to $L2$.

Results of the pixel-based comparison can be either used directly to generate the matching cost or they can be improved by additional methods. Examples of them are the Earth Mover Distance [130], which is a measure of the minimum amount of "work" required to transform one probability distribution into another, and Kernel Density Estimation [119], which is a non-parametric method for estimating the probability density function.

Window methods

The second group of image similarity measurement methods based on the comparison of the pixels within a certain window around each pixel of the images. Being less computationally efficient than the pixel-based methods, this group tends to provide higher accuracy results due to the higher number of comparisons per a certain pixel.

For every 2D pixel in image I with coordinates (u, v) the window $D \in \mathbb{Z}^2$ defines the set of shifts of the coordinate of this pixel. The common window-based methods are [54]:

- Sum of absolute differences, defined as:

$$SAD(I_1, I_2, D) = \sum_{[i,j] \in D} |I_1(u, v) - I_2(u + i, v + j)| \tag{5.3}$$

- Sum of squared differences, defined as:

$$SSD(I_1, I_2, D) = \sum_{[i,j] \in D} (I_1(u, v) - I_2(u + i, v + j))^2 \tag{5.4}$$

- Normalized cross-correlation, defined as:

$$NCC(I_1, I_2, D) = \frac{\sum_{[i,j] \in D} (I_1(u, v) - \bar{I_1})(I_2(u + i, v + j) - \bar{I_2})}{\sqrt{\sum_{[i,j] \in D} (I_1(u, v) - \bar{I_1})^2 \sum_{[i,j] \in D} (I_2(u + i, v + j) - \bar{I_2})^2}}, \tag{5.5}$$

where $\bar{I}_1$ and $\bar{I}_2$ stand for the average value within the image and the search window.

All previously listed methods for measuring the similarity between images lack the important detail, which is crucial for real-world image processing. They are sensitive to the illumination changes between different images, which means that they can't provide accurate similarity estimation if the camera sensors providing images with different brightness and contrast. Also, these methods tend to fail in case of even slightly different focusing of the lenses of images capturing devices.

Trying to compensate for these effects, Zabih and Woodfill proposed two non-parametric (*i.e.* independent on the actual pixel values) methods for the image similarity measurement in [178], namely rank and census transform. First method forms a structure, which keeps the sorted radiance values around the reference pixel in ascending order and assigns the reference pixel rank based on the corresponding value position in this structure. We will focus on the second method in more detail, as it forms the main part of the algorithm used in this dissertation to compute depth maps.

Census transform

Census transform is an image transformation that takes place as a modification of the rank transform by binary counting of the radiance value relations between the reference pixel and the surrounding pixels within a particular window D. Every pixel of census-transformed image I_c stores the so-called bit-string of the comparison result. For the source image I it is defined as follows:

$$I_c(u,v) = \bigotimes_{[i,j]\in D} \xi(I(u,v),\, I(u+i,v+j)), \tag{5.6}$$

where $\otimes$ stands for bit-wise concatenation. ξ defines the pixels comparison function, which finds if the value of the pixel within the window is greater or lower than the reference:

$$\xi(p_1,p_2) = \begin{cases} 0, & p_1 \leqslant p_2 \\ 1, & p_1 > p_2 \end{cases}. \tag{5.7}$$

This transform is somewhat similar to the local binary patterns concept [47]. The main difference is that the census transform is defined densely, *i.e.* for the entire image, while local binary patterns, as their name implies, are defined sparsely.

The choice of a window for the census transform is another interesting subject. Since the bit-string is formed as its output, the natural desire would be to use a number of window elements that is a multiple of or close to a multiple of 8 (*e.g.* 8, 16, 32). The reason for this is that the subsequent computation of the coincidence between the two images is done by exclusive disjunction, which can be effectively implemented on modern hardware by using a single processor instruction, as will be discussed in the chapter 6.

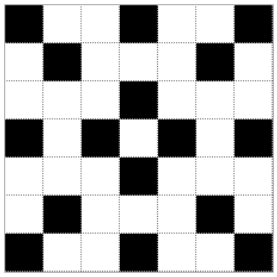


FIGURE 5.1: An example of sparse pattern for census transform within 7x7 window. Black pixels represent assigned pixels for census transform

Another motivation for using the sparse pattern is related to the reduction of the computational time for the census transform. An example of the sparse census transform pattern is shown on Fig. 5.1.

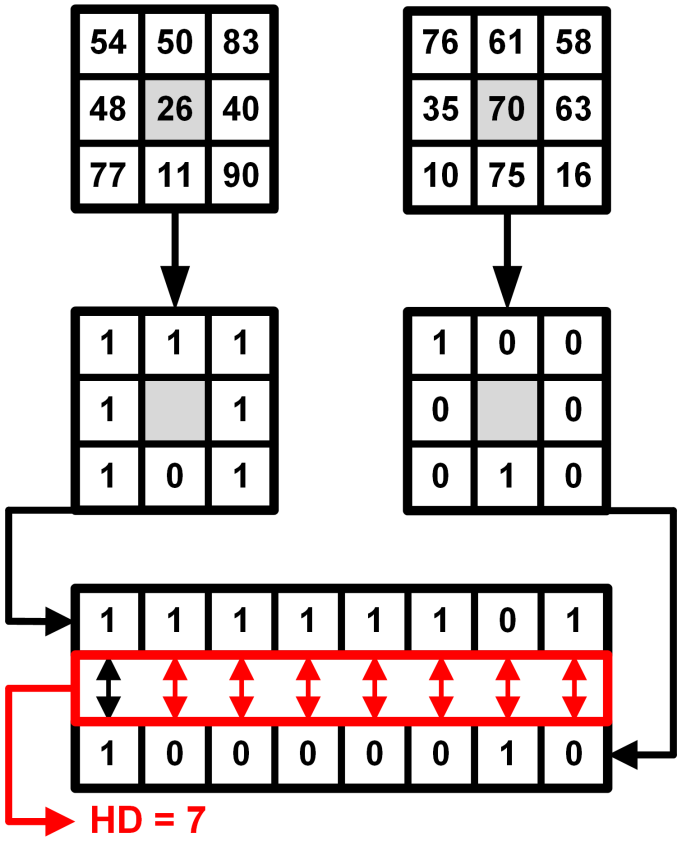


FIGURE 5.2: Visualization of census transform and Hamming distance

The interesting feature of the census transform is the ease of implementation of the comparison of the two census-transformed images. It is based on estimating the similarity between two strings of bits of the same length, which is based on counting the number of corresponding non-matching pairs of bits in this strings. This estimation is called Hamming distance [90]. For two bit strings from census-transformed images, represented as two vectors

$x_i, x_j \in \mathbb{Z}_2^n$, the Hamming distance HD is estimated as a quantity of elements with different values:

$$HD(x_i, x_j) = \sum_{k=1}^{n} x_{ik} \oplus x_{jk}, \tag{5.8}$$

where $\oplus$ denotes exclusive disjunction.

Fig. 5.2 visualizes the process of finding the census transform of the image and the Hamming distance estimation. The next section describes how it is used to build a matching cost, which will be further utilized to compute the depth map.

5.1.2 Matching cost construction

A term "matching cost" correspond to the structure, which contains the results of the similarity comparison between two images [1]. A visualization of the matching cost is shown on Fig. 5.3. It is typically 3D, having the two

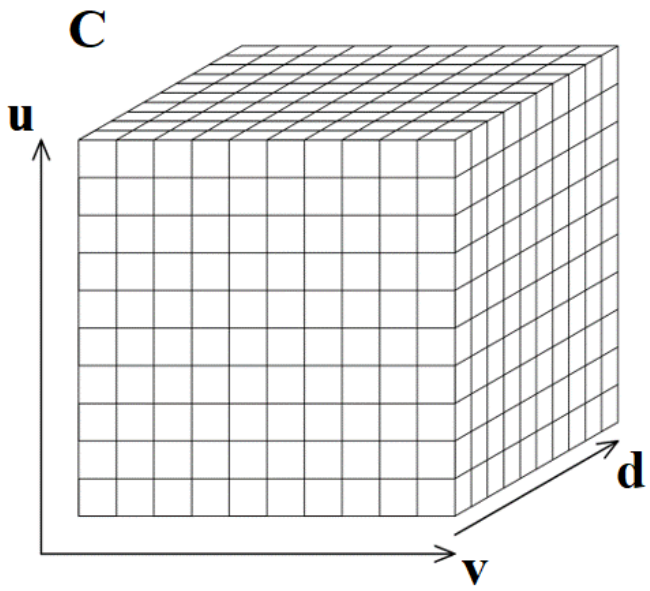


FIGURE 5.3: Visualization of the matching cost

dimensions corresponding two the image dimensions (axes y, x on Fig. 5.3), and the third related to the important concept for 3D reconstruction named "disparity range" (axis d). This concept is inextricably linked to the definitions of disparity, *i.e.* the offset of a pixel in one image relative to another, and the disparity hypothesis, *i.e.* the assumption of a certain existing value of the offset. Thus, we can define the disparity range as the range of all possible disparity hypotheses that will be tested by comparing one image (reference view) and the pixels of another image with a given offset. From this we can

[1] In the past, similar structure was called "disparity-space image" [64]; within the scope of the dissertation we will refer to it as to the "matching cost".

define matching cost as a totality of all tested disparity hypotheses, each element of which, in the case of two images I_1, I_2, placed on the same horizontal axis with aligned camera centers, is defined as:

$$C(u, v, d) = CMP(I_1(u, v), I_2(u, v + d)), \tag{5.9}$$

where $CMP()$ denotes the generic comparison function. Depending on the type of images and the chosen method, this can be any function from Eq. 5.1-5.5, or Eq. 5.8.

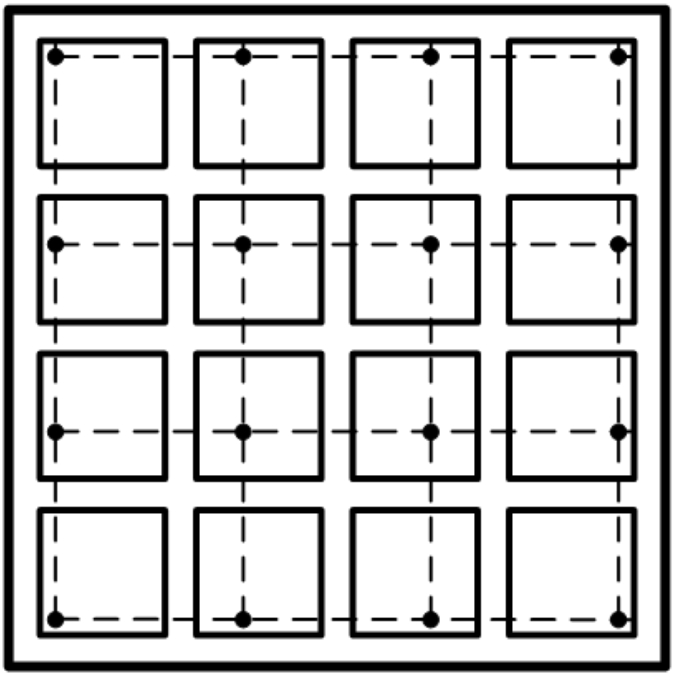


FIGURE 5.4: Visualization of pixel relation in LF

Construction of the matching cost based on the LF images can be derived from the Eq. 5.9 due to the similarities stated in Section 2.3.3. For that we will use the previously mentioned two-plane parameterization, discussed in Section 2.3.2. For the LF, denoted in Eq. 2.19 as L, defining the reference view as $(\hat{s}, \hat{t})$ and a certain disparity hypothesis d, a pixel position, which matches this hypothesis in another view (u, v) can be determined as:

$$\hat{p}(u, v, s, t, d) = L(u + (\hat{s} - s)d, v + (\hat{t} - t)d, s, t) \tag{5.10}$$

This relation is illustrated in Fig. 5.4.

Thus, for a particular pair of LF views the similarity can be measured as

$$CMP(L(u, v, s, t), \hat{p}(u, v, s, t, d)) \tag{5.11}$$

Speaking of LF, the obvious approach might seem to be the using 4D of matching cost, since such a structure can store the results of N images comparisons in itself, introducing a channel with $N - 1$ comparisons besides the discussed three. However, the regression of such a matching cost, especially

in the case of large N, seems to be a problem for which, firstly, proportionally more memory costs are assumed to store such a matching cost, and for which, secondly, proportionally more time should be dedicated to computations. It is obvious that in the context of real-time embedded processing requirements such demands are not feasible, which leads to the use of 3D matching cost.

Thereby, the generic matching cost from the LF is formed as:

$$C(u,v,d) = \sum_{s,t \in L} CMP(L(u,v,s,t), \hat{p}(u,v,s,t,d)) \tag{5.12}$$

Disparity regression The resulting matching cost can be used for the subsequent disparity map regression, which can be estimated using the winner-takes-all (WTA) strategy. In this strategy, the disparity value with the lowest cost is chosen as the winner for each pixel of the disparity map D as:

$$D(u,v) = \arg\min_{d} C(u,v,d). \tag{5.13}$$

The WTA strategy is efficient because it only requires finding the minimum cost among the subset of the matching cost. However, a big problem for real-time estimation is the depth of this matching cost along the d-axis. The time it takes to regress the disparity map is directly proportional to the depth of this matching cost. Questioning how this time could be reduced, we came up with the idea of matching cost bordering, which will be described later.

Matching cost bordering

There are several ways to reduce the calculation time of the matching cost estimation and processing. The naïve approach involves reducing the image size, which leads to a reduction of the number of pixels given for processing. However, it potentially leads to the problem of decreasing computation accuracy. Reducing the number of views greatly affects accuracy in small details and also poses the additional task of upsampling depth maps later, which in turn may introduce additional latency.

Another potential method is to use just a part of the provided LF views. It will reduce the accuracy of the disparity map, again especially on fine details, but would still keep the disparity of the majority of the scene object in the values close to the actual disparity. With that in mind, we decided to generate bordering information of the matching cost by first computing the disparity map on the full disparity range, but involving only a fraction of LF views, which further be used as a source of disparity range limitation during the estimations on the full amount of LF views.

In order to create boundary values for correspondence search in whole LF space by more computationally-intensive algorithm the initial disparity map is calculated by using lower computationally-intensive algorithm. Calculations in this step are simplified compared to whole LF space correspondence matching by reducing the number of processed views and preserving

changes only in one angular direction. In other words, by bordering we mean limiting the range of disparity hypotheses along the d-axis.

During the work on this dissertation, we developed a strategy for generating the initial disparity map used to constrain the matching cost. For that, we used four LF views, equidistant from each other on the two axes formed by the reference view positions, named "anchors". In other words, as anchors we define the views at the borders of the LF, which are lying on the cross with the reference view in its center. Fig. 5.5 illustrates how the reference and anchor views are placed in the LF space.

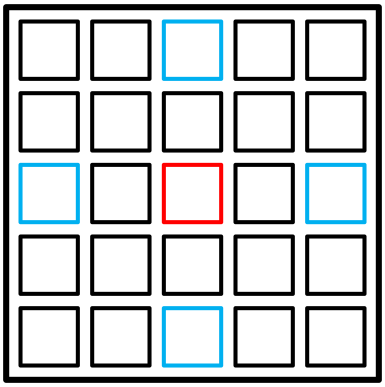


FIGURE 5.5: Reference (red) and anchor (blue) views of the LF

These views are important for the disparity estimation as they cover all visible scene points, projected to the LF image. It also allows to cover the maximum possible disparity range, which can be captured by the LF. In the following, these "anchors" will be denoted as I_L, I_R, I_T, I_B as left, right, top and bottom views respectively. Coordinates of this views *w.r.t.* to the reference view $(\hat{s}, \hat{t})$ can be set as $\{(\hat{s}, 0), (\hat{s}, t_{max}), (0, \hat{t}), (s_{max}, \hat{t})\}$, where s_{max} and t_{max} correspond to horizontal and vertical angular dimensions of the LF.

First, we build four matching costs based on pairs of $I_L - -I_R$, $I_R - -I_L$, $I_T - -I_B$, $I_B - -I_T$. By applying the Eq. 5.13 to the matching costs, a set of disparity maps $V = \{D_{LR}, D_{RL}, D_{TB}, D_{BT}\}$ is obtained. These disparity maps are estimated in the coordinate system of the respective view, and for their utilization as a source for the bordering information they need to be re-projected to the coordinate system of the reference view.

For doing so, we can use the relation, described in Eq. 5.10. By changing the terms, related to the order of the LF view from $(\hat{s} - s), (\hat{t} - t)$ to $(s - \hat{s}), (t - \hat{t})$ we can re-project the depth values from the local coordinate systems of the specific LF view to the coordinate system of the reference view. It forms a set of re-projected disparity maps $V_R = \{\hat{D}_{LR}, \hat{D}_{RL}, \hat{D}_{TB}, \hat{D}_{BT}\}$, where every map is stored *w.r.t.* pixel coordinates in the reference view.

In order to filter away the possible wrongly estimated disparity values we apply a disparity consistency verification step to every disparity map in V_R. Since disparity maps, which computed from the images placed on one axis, may contain the information about the scene points, which are not visible by other pairs from another axis, it makes sense to do the consistency

verification pairwise using at first only images which are placed on one axis, so the potential invisible areas on the other disparity maps won't be affected by filtering. For that, for every pair $(\hat{D}_{LR}, \hat{D}_{RL})$ and $(\hat{D}_{TB}, \hat{D}_{BT})$ we check if the per-pixel difference of the pixels as:

$$|D_{V_1}(u,v) - D_{V_2}(u,v)| < \varphi, \tag{5.14}$$

where D_{V_1}, D_{V_2} are the pair of verified disparity maps, φ stands for confidence threshold. It forms two confidence maps, one per each pair, based on which certain pixels can be excluded from the process of disparity map fusion.

$$CMT(u,v) = \begin{cases} 1, & |D_{V_1}(u,v) - D_{V_2}(u,v)| < \varphi, \\ 0, & \text{otherwise} \end{cases} \tag{5.15}$$

Fusion of disparity maps is done as the average of the corresponding pixel values from V_R, respecting their validity in the corresponding confidence maps, resulting in the fused disparity map D_F.

The disparity map thus obtained suffers from the following disadvantages:

- Due to the limited number of images used for the estimation, there will be noise on the result, which may then be directly transmitted to the final disparity map, and thus negatively affect its accuracy.
- There will be empty areas on the image, the presence of which negatively affects the execution time, as in areas without certain values it is impossible to form any hypotheses about their sub-disparity range.

In the earlier publications, we used a single- or double-pass median filter on D_F to partially compensate for the first effect. Over time, we have formed a new strategy related to the consequent use of per-layer disparity filtering and holes filling, which will be described below.

Per-layer disparity filtering This filter works as follows. For every available disparity hypothesis $d \in T$ the new image, containing only pixels with disparity value d is created:

$$D_{C_d}(u,v) = \begin{cases} D_C(u,v), & D_C(u,v) = d \\ 0, & D_C(u,v) \neq d \end{cases}. \tag{5.16}$$

This image is a subject for the morphological closing operation [44], which stands for erosion, followed by dilation. Filtered disparity maps are combined back with preserving the presence of already associated pixels by processing the decomposed disparity maps from far to near. The resulting set of images is combined to the disparity map, used for the further borders generation step.

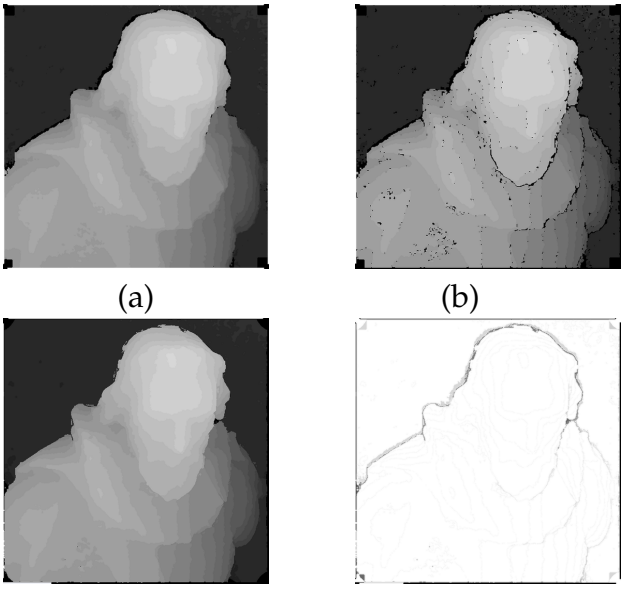


FIGURE 5.6: Initial disparity post-processing steps: (a) initial disparity map, (b) per-layer disparity filtering, (c) holes filling, (c) difference between (a) and (c)

Holes filling After consistency check, merging and following filtering the disparity map has considerable amount of pixels without associated disparity value. In order to reduce the number of such pixels we use a holes filling technique based on the neighborhood pixel information and color consistency. For a missing disparity pixel the filling procedure is based on the median value of nearby values within a window of a certain size.

For the pixels, placed on the edges, such filling can lead to associating false values. To prevent it we use values from a corresponding color LF view. Color pixels in the window are checked for their Euclidean distance from the reference pixel being below the threshold, based on which they are involved in the median value estimation.

This algorithm is performed iteratively, and the stopping criteria of this method are defined as the number of iterations and the number of non-empty pixels. To prevent jamming of the method on the same pixels after the third iteration the window size is increased logarithmically and the threshold is relaxed accordingly. Fig. 5.6 demonstrates the application of these steps to the initial disparity map. Using these two algorithms positively affected the edges [142].

Bordering information generation Calculation of disparity maps based on two LF views allows to sufficiently determine if not close to exact values for subsequent calculation of disparity maps from many LF views, then at least reference values, relative to which sub-disparity ranges can be constructed. The initial disparity map serves for creation of computational limitation for disparity hypothesis range. It fulfills two purposes. First, the generation of

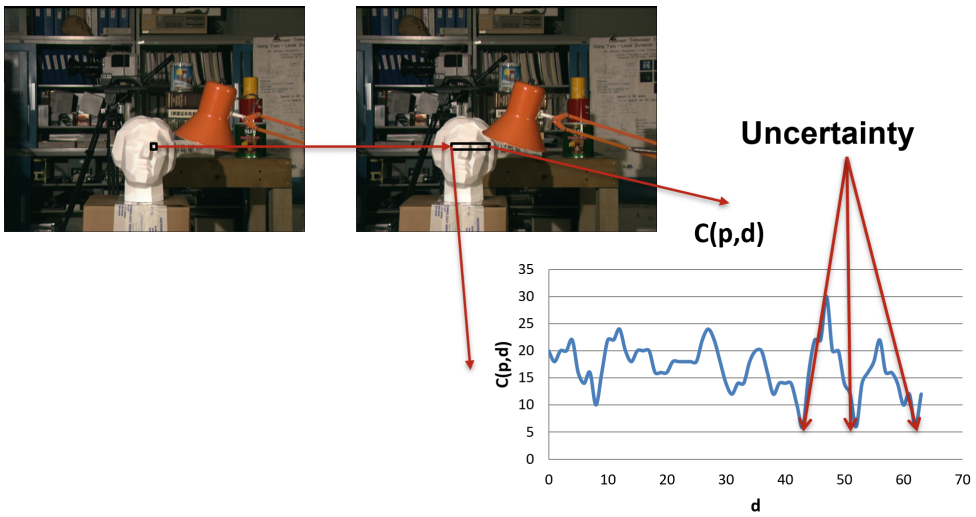


FIGURE 5.7: Visualization of the local matching cost estimation problem: uncertainty in the minimal value

matching cost from many LF views is a time-consuming task. Limitation of the disparity search range by the bordering information reduces the running time of the matching procedure. Second, due to the ambiguities from the matching cost estimation the wrong estimations and noise pixels can be present in final disparity map. Bordering information prevents appearance of these issues.

D_F with the applied per-layer disparity filtering and holes filling is used for generation of boundaries for the further estimation. These boundaries will limit matching cost generation in the whole LF space. Two structures named high and low borders (D_H and D_L respectively) are generated by using the border threshold λ in such a manner:

$$D_H(u,v) = D_F(u,v) + \lambda; D_L(u,v) = D_F(u,v) - \lambda. \tag{5.17}$$

The values, which lie outside of predefined disparity range ($D_H > d_{max}$, $D_L < d_{min}$) are saturated accordingly. Invalid values from D_C are marked in the corresponding borders for re-computation on the whole disparity range T.

In the end, to calculate the final matching cost using all the images in LF these boundaries determine for which specific disparity hypotheses this matching cost will be calculated. However, the accuracy of a disparity map regression derived only from the matching cost compiled in this way leaves much to be desired, since the obtained result may still not be smooth enough and have some noise in it. To prevent this there are methods for optimizing the matching cost, which will be discussed in the following section.

5.1.3 Optimization methods

The need for matching cost optimization methods stems from the problem of having several local minima in different parts of the matching cost. In

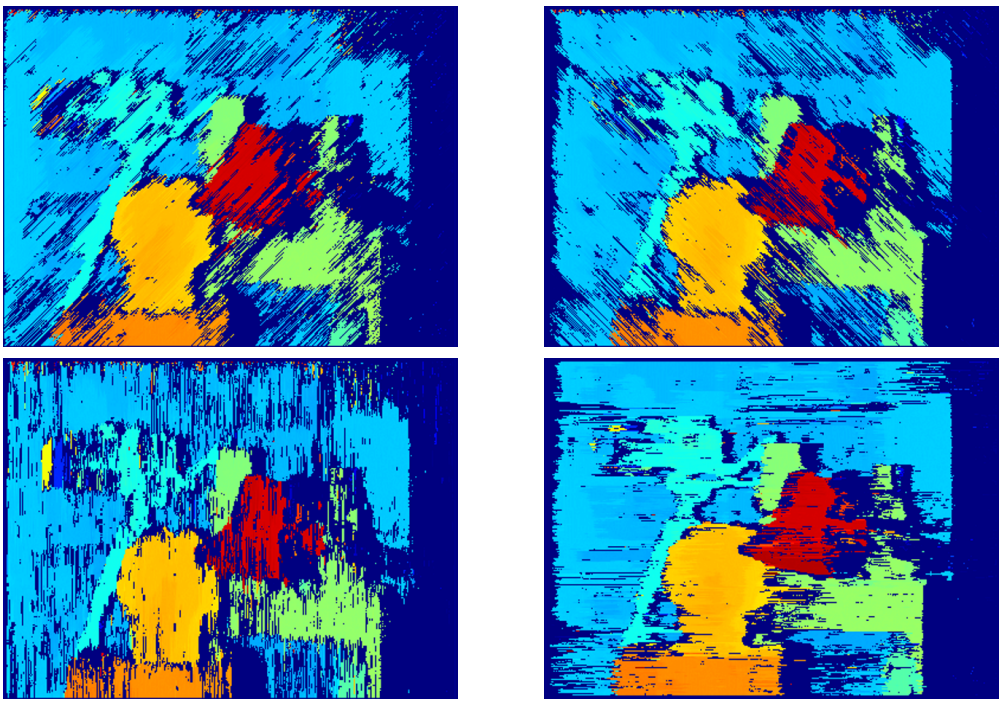


FIGURE 5.8: Examples of the disparity maps, regressed from individual SGM paths

computations based on so-called local methods, *i.e.* the methods we discussed earlier, which are based only on the similarity of pixel-by-pixel or window-by-window values, the acquired values usually may be insufficient for a correct decision on a particular disparity value based on the Eq. 5.13. It is demonstrated on Fig. 5.7.

A potential method for getting rid of these kinds of problems is to use so-called global optimization techniques. Typically, such methods involve optimizing an energy function, when all pixels are taken in the account, in the way that balances the trade-off between the data term, which shows how well the estimated disparity suits the observed data, and smoothness term, which verifies how smooth and consistent the disparity map is. Popular examples of such algorithms are the graph cuts, proposed by Boykov *et al.* in [21], and belief propagation from Sun *et al.* [144].

While the disparity maps obtained by such methods are quite accurate, the big problem of these methods is the computation time. Even on the modern hardware, the computations of maps with these methods take such time intervals that do not allow such algorithms to be classified as real-time. However, there is a technique that combines the merits of both local and global groups of the methods, being relatively fast, but also quite accurate, since it does not perform matching cost optimization on the entire image space, but on relatively small paths around each pixel. This method is named SGM and will be explained below.

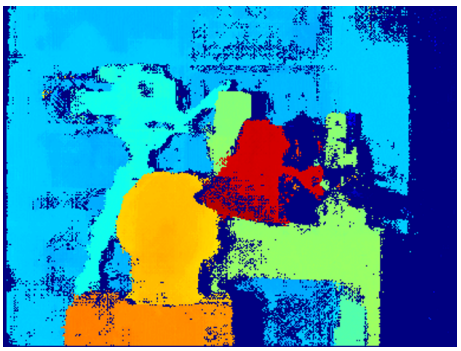


FIGURE 5.9: Disparity map, obtained by summarizing the matching costs among different paths, based on Fig. 5.8

Semi-Global Matching

The SGM was proposed by Hirschmüller in [52] as an attempt to replace the bulky global cost functions for the stereo reconstruction task. This method can be considered as the optimal one between local-only matching cost collection and the global cost optimization, which can provide the most accurate result, but with significant computational load.

The idea behind SGM is to use the local matching with partially (semi-) global optimization of the cost matching. Having the collected matching cost, this algorithm performs cost aggregation on different traversing directions in the matching cost space using a cost aggregation function. SGM considers multiple paths in different directions, such as left-to-right, right-to-left, top-to-bottom, bottom-to-top, and diagonal directions. This algorithm uses two types of penalties during the aggregation: *P1* for the neighbor disparities and *P2* for the "far" disparities.

The path-wise aggregation for each pixel $p = (u, v)$ and depth hypothesis d in a predefined range, after traversing in direction r, formulated as a 2D vector with the coordinate of a pixel traversing $r = \{\Delta u, \Delta v\}$, aggregated cost L_r is:

$$\begin{aligned} L_r(p,d) &= C(p,d) + \\ &\min\big(L_r(p-r,d), \\ &L_r(p-r,d-1) + P1, \\ &L_r(p-r,d+1) + P1, \\ &\min_t L_r(p-r,t) + P2\big), \end{aligned} \tag{5.18}$$

where *P1* and *P2* are penalty parameters, $P2 \geqslant P1$. Fig. 5.8 shows the result of disparity regression from individual traversing paths with visible visual artifacts.

Traversed costs are then summarized through all traversing directions:

$$C_s(p,d) = \sum_r L_r(p,d). \tag{5.19}$$

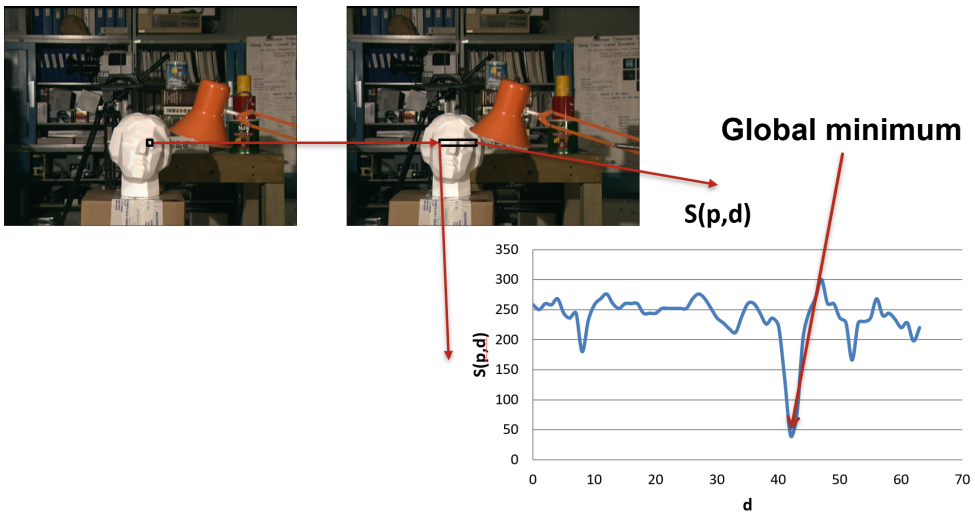


FIGURE 5.10: Visualization of the fragment of matching cost after applying SGM on it.

Fig. 5.9 shows how the result of the disparity regression of C_s may look like.

These equations are used on the previously generated matching cost by Eq. 5.9 before applying Eq. 5.13 to it.

Fig. 5.10 demonstrates, how the SGM solves the global minima search problem.

In our implementation, SGM is performed on the full disparity scale for the case of initial disparity map estimation, and only on the disparity values, which are in the predefined bounded range $d \in [D_L(p), D_H(p)]$, for the final disparity map, so values outside the boundaries do not affect the aggregation process.

Accuracy of the disparity map after SGM is highly improved. However, it is possible to further improve the accuracy of the computations by refining the disparity with additional sub-pixel-level estimates, which will be explained later.

5.1.4 Sub-pixel refinement

In general, matching cost stores the data for each particular disparity hypothesis. Applying the straightforward approach, described by Eq.5.13, we can generate a disparity map in integer values of displacements. Sub-pixel estimates refer to computations that are performed within a range of less than one pixel. These calculations are based on the corresponding matching cost data, so that a more accurate and complete matching cost allows for more precise estimations.

Classically, for sub-pixel interpolation of the disparity d value at a point (u, v), three values of matching cost $C(u, v, d-1), C(u, v, d), C(u, v, d+1)$ are used. One of the most common methods for sub-pixel disparity estimations is parabolic interpolation [134]. Geometrically, it can be visualized as a parabola, which goes through the three points on a coordinate system, where the abscissa axis denoted the values of disparity, and corresponding values

of matching cost are defined by their coordinate on the ordinate axis. The minimum value of the parabola fitted to these three points is the sub-pixel value for the given pixel. This principle is visualized on Fig. 5.11.

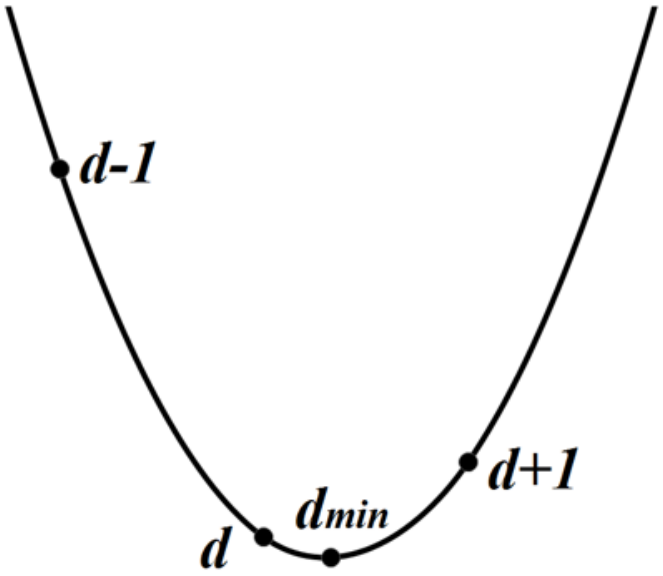


FIGURE 5.11: Visualization of the parabolic interpolation

For each pixel (u, v) in interpolated disparity map D_N procedure for calculation of the interpolated value based on previously regressed from C_s disparity map D can be expressed as follows:

$$D_N(u,v) = D(u,v) + \frac{C_S(u,v,d-1) - C_S(u,v,d+1)}{2\,(2\,C_S(u,v,d) - C_S(u,v,d-1) - C_S(u,v,d+1))} \tag{5.20}$$

The disadvantage of parabola fitting is the uneven distribution of the resulting sub-pixel results. It potentially can lead to higher level of errors on the sub-pixel resolution.

To avoid this, Haller and Nedevschi proposed Symmetric-V interpolation scheme in [42]. For every pixel (u, v) the values of interpolated image D_N are computed as:

$$\begin{gathered} D_N(u,v) = D(u,v) + \\ \begin{cases} + \left(0.5 - 0.25 \left(\frac{(M3-M1)^2}{(M2-M1)^2} + \frac{(M3-M1)}{(M2-M1)}\right)\right); M2 > M3 \\ - \left(0.5 - 0.25 \left(\frac{(M2-M1)^2}{(M3-M1)^2} + \frac{(M2-M1)}{(M3-M1)}\right)\right); M2 \leqslant M3 \end{cases} \\ M1 = C_S(u,v,d), M2 = C_S(u,v,d-1), M3 = C_S(u,v,d+1) \end{gathered} \tag{5.21}$$

In the case of the algorithm, presented in this dissertation, due to the bordered matching cost, interpolation can only be performed on pixels, in which disparity value $D(u,v) \in [D_L(u,v)+1, D_H(u,v)-1]$ and $|[D_L(u,v)+1, D_H(u,v)-1]| \geqslant 3$. If the disparity value does not satisfy these conditions,

then $D_N(u,v) = D(u,v)$. Potentially, this could also lead to some loss of accuracy; however, compared to the merits obtained using matching cost bordering, we believe that these errors can be neglected.

5.1.5 Disparity-to-depth conversion

Disparity-to-depth conversion is performed by a classical equation, based on the focal length f and the baseline b between two cameras on one axis at the maximum distance. The principles of LF parameterization allows to do it in the way:

$$D_Z(u,v) = \frac{fb}{D_I(u,v)}. \tag{5.22}$$

The obtained result can be further improved by performing computations when displacing a reference view, as will be discussed later.

5.1.6 Point Cloud extension

Choosing another reference view for further computations aims to make calculations with respect to those areas of the LF that were not visible from the central reference view.

The disparity map, described in the previous section, is further used as initialization for the point cloud, which will be optimized using all LF views. For the point cloud generation the disparity map D_I needs to be converted to the depth map, based on the common focal length of LF views f and the distance between two adjacent views on one axis b, which remains same for all view pairs based on the LF parameterization, by applying the Eq. 5.22.

Depth values from D_Z are used for getting the 3D points P as:

$$\begin{gathered} P = [XYZ]; Z = D_Z(u,v) \\ X = Z\frac{v}{f}; Y = Z\frac{u}{f} \end{gathered} \tag{5.23}$$

To include the information for scene points, which are not visible in the reference LF view, we define the reference views in the corners of the LF. The initial disparity map, which was subjected for the disparity bordering ranges estimation, is reprojected to the new reference view coordinates by using Eq. 5.10. The reprojected image is then used for the generation of bordering information for the further estimation of final disparity map for the new reference view based on the cross-lying views, repeating the previously discussed steps.

The point clouds are obtained by applying Eq. 5.22 and 5.23 and transposed to the viewpoint of the original reference view. For multiple point cloud registration we use classical iterative closest points (ICP) approach [9]. While ICP defines the needed transformation for all points, we remove the already preserved ones from the new point clouds by their projection to the

original reference view plane and deleting the matching points. It allows to construct the joined point cloud by just combining the point sets.

For optimization purposes the information about point origin is stored alongside with the point. Every point cloud is separately projected to various LF viewpoints. By that, we are trying to find value of Z, which minimizes the error between the projected and presented pixel in LF views:

$$\sum_{s=1}^{|s|} \|\hat{p}(u,v,s,\hat{t},d) - p(u,v,s,\hat{t},d)\| + \sum_{t=1}^{|t|} \|\hat{p}(u,v,\hat{s},t,d) - p(u,v,\hat{s},t,d)\|, \tag{5.24}$$

where $\hat{p}$ is the projected pixel, estimated by the principles from Eq. 5.23. Based on the point origin it is optimized only on the frames, where the point is visible. For simplification we define the possible configuration of viewpoints based on the cross-lying LF views.

5.1.7 Post-processing techniques

One of the most popular technique for the disparity map filtering is applying the median filter on them. It is an edge-preserving technique, meaning that it keeps original edges, which is especially important for the disparity maps. This method replaces the pixel values of the disparity map by the median value of pixels around it within a certain window, typically 3×3 pixels. It helps to remove the impulse noise from the disparity map.

Output on the non-linear optimization step contains some specific noise, which requires additional post-processing efforts. For reducing such noise we found that combined bilateral filter, proposed by Wasenmüller *et al.* in [167], suits best. It composes the classical bilateral filter with joint bilateral filter, published in [79].

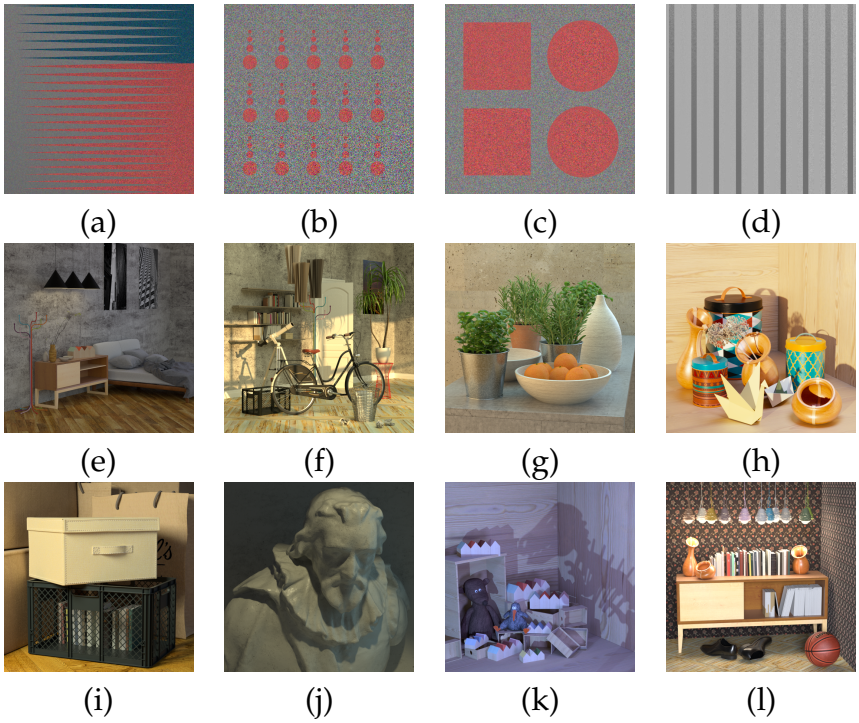


FIGURE 5.12: Center images of LFs from [55]: first row – "stratified" scenes, middle row – "testing" scenes, last row – "training" scenes; (a) "backgammon", (b) "dots", (c) "pyramids", (d) "stripes", (e) "bedroom", (f) "bicycle", (g) "herbs", (h) "origami", (i) "boxes", (j) "cotton", (k) "dino", (l) "sideboard"

5.2 Evaluation

This quantitative evaluation is done based on three versions of the algorithm developed at different times within the scope of this dissertation, namely BSL [6], FSL [8] and PSL [5]. The role and purpose of each of the algorithms can be described as follows

- BSL aims to be the first algorithm to solve the problem of rapid disparity estimation from LFs, on which the basic ideas for subsequent real-time optimization can be tested.
- FSL was the forerunner of real-time algorithm for real-world embedded system described in the Chapter 6, in which shortcomings of the previous method were taken into account and experiments for optimizations related to fast data processing were carefully performed.
- PSL, despite being a slower version, tested what algorithmic improvements could be made in the future with access to more productive embedded hardware, while remaining a minimalist algorithm in terms of number of operations performed.

The comparison of these specific features of these algorithms is presented in Table 5.1.

Distinctive features	BSL [6]	FSL [8]	PSL [5]
Window for census transform	Sparse, 7×7	Sparse, 7×7	Dense, 7×9
SGM traversing directions	16	16	8
SGM on initial disparity map	✓	✓	✓
SGM on final disparity map	✗	✓	✓
Initial disparity map re-projection	✓	✗	✓
Per-layer disparity filtering	✗	✗	✓
Holes filling	✗	✗	✓
Interpolation	Parabolic	Parabolic	Symmetric-V
Interpolation on initial disparity map	✓	✗	✗
Point cloud processing	✗	✗	✓
Combined bilateral filtering	✗	✗	✓
Confidence threshold φ (Eq. 5.14)	3	3	2
Border threshold λ (Eq. 5.17)	2	2	1

TABLE 5.1: Comparison of the configurations of different algorithms used in the dissertation

5.2.1 Dataset

We use the LF images, provided by Honauer *et al.* [55] through 4D LF Benchmark. 12 synthetic scenes are provided for the main evaluation; each scene is represented by the 9×9 LF, composed from 8-bit RGB images with resolution of 512×512 pixels. For every image a 512×512 disparity map with sub-pixel disparity resolution is provided.

Scenes are grouped in three categories: "training" for evaluation and parameters adjustment, "stratified" with special challenging cases, and "test" for "blind" verification. Camera settings and disparity ranges provided for every LF, high resolution disparity and depth maps are provided only for "training" and "stratified" datasets. In this section we present image result comparison for "dino" scene; results for other scenes can be found at [1] under corresponding acronyms.

5.2.2 Metrics

Several different metrics for the quality of the result of the different algorithms are provided within the benchmark.

The classical measurement for finding how far the algorithm output d from the ground truth data gt is MSE, which for the certain pixel mask m, computed *w.r.t.* reference view, can be estimated as [55]:

$$MSE(d, gt)_m = \sum_{(u,v)\in m} (d(u,v) - gt(u,v))^2 \frac{100}{|m|} \tag{5.25}$$

Another metric presented in the benchmark named *BadPix* and stands for the percentage of the properly estimated pixels, for which the difference between them and the corresponding ground truth is below a certain threshold T (here

$T = 0.07$):

$$BadPix(d, gt, T)_m = \frac{\{(u,v) \in m : ||d(u,v) - gt(u,v)| > T|\}}{|m|} \tag{5.26}$$

The last quality metric $Q25$ is estimated as the maximum absolute disparity error of the best 25% of pixels.

The classic metric for not the quality, but the performance of an algorithm is the basic measurement of the execution time. On top of that, for purposes of interpreting our result in terms of processing time-driven contributions, in part related to matching cost bordering, we propose a metric M, which stands for percentage of correctly computed pixels per second, formulated as

$$M = \frac{100\% - BadPix}{Runtime}\left(\frac{\%}{sec.}\right). \tag{5.27}$$

The benchmark provides additional photo-consistency metrics, *e.g.* surface smoothness; these metrics are not presented within the dissertation and are available on the benchmark website [1].

5.2.3 Parameters

The parameters of all the algorithms were optimized for the best possible result on *BadPix* metric.

We were trying different comparison functions for the matching cost estimation and empirically came up to the following configuration. For the initial disparity map estimation we were using the matching cost obtained by the Hamming distance estimation with Eq 5.8 between census-transformed images. Estimation of final disparity map is done by L_2-based matching cost using Eq. 5.2.

For the census transform different patterns of the aggregation window have been evaluated. For the experiments in BSL and FSL we use 7×7 sparse pattern, shown on Fig. 5.1. Another options for sparse Census window are listed in [94].

Experiments on 7×9 dense census pattern (as the closest to 64-bit word) were conducted in PSL. While it covers bigger image area and potentially leads to the higher accuracy, necessity of such a pattern in real-time configuration is questionable due to the potential overhead created by estimation of transform within relatively big window.

Penalty parameters for the SGM *P1* and *P2* were set to 21 and 45 for BSL and FSL configuration, and adjusted to 20 and 40 respectively for PSL for the final disparity map estimation. Due to the different matching cost values resolution in case of census-based matching cost, we set these parameters to 30 and 150 for the initial disparity map estimation on PSL and to 17 and 35 on FSL. Number of disparities hypothesis was adjusted accordingly to the data, provided in configuration files for each of the scene.

In PSL, holes filling algorithm uses 25 iterations as the stopping criteria of the optimization. Initial window size is set to 5 and initial threshold for

	BadPix, %	
	Median	Average
EPI1 [70]	22.89	24.32
EPI2 [164]	22.94	22.65
EPINET [138]	3.38	4.93
FASTLFNET [62]	8.24	9.07
LF [66]	16.15	16.19
LFOCC [160]	18.45	17.58
OFSY [143]	11.33	12.04
RM3DE [108]	7.99	10.22
RPRF [61]	9.89	10.02
SCGC [139]	10.21	14.3
SPO [183]	8.78	8.47
BSL [6]	13.41	12.74
FSL [8]	11.92	12.95
PSL [5]	11.61	12.79

TABLE 5.2: Evaluation of different algorithms with $BadPix(0.07)$ metric on 4D LF Benchmark [55]. Italics indicate articles written in scope of this dissertation.

the distance between color values of the pixels is set to 5. Standard deviation values of 0.5 and 2.5 were set for combined bilateral filter [167].

5.2.4 Results

Tables 5.2-5.6 present the evaluation of the subset of the algorithms on the aforementioned metrics. We provide a comparison of the proposed algorithms with the state-of-the-art methods, presented in Section 3.2: EPI1 [70], EPI2 [164], EPINET [138], FASTLFNET [62], LF [66], LFOCC [160], OFSY [143], RM3DE [108], RPRF [61], SCGC [139], SPO [183]. Additionally, Fig. 5.13 visualizes the differences in values of metrics for BSL, FSL and PSL variants.

The results of our developed algorithms at the time of publication were in the middle position relative to the other algorithms on the BadPix metric, which was our main metric for evaluating algorithm quality, as it is the basis of the proposed M-metric. A similar conclusion can be drawn from an analysis of the Q25 results in the Table 5.4.

While the results does not change drastically between proposed algorithms on $BadPix$, the higher change is noticeable while comparing them on MSE (Table 5.3). All the additional functionality of FSL allows to outperform most of the classical algorithms on this metric, losing out to deep learning methods. However, the advantage of our approaches is that there is no need

	MSE	
	Median	Average
EPI1 [70]	3.93	5.98
EPI2 [164]	5.72	8.24
EPINET [138]	1.21	2.48
FASTLFNET [62]	1.61	2.46
LF [66]	7.96	9.13
LFOCC [160]	2.80	6.69
OFSY [143]	5.43	7.03
RM3DE [108]	1.46	3.92
RPRF [61]	3.76	5.68
SCGC [139]	3.94	6.58
SPO [183]	3.31	3.97
BSL [6]	5.43	7.28
FSL [8]	3.97	6.64
PSL [5]	2.78	5.14

TABLE 5.3: Evaluation of different algorithms with MSE metric on 4D LF Benchmark [55]. Italics indicate articles written in scope of this dissertation.

	*Q*25	
	Median	Average
EPI1 [70]	1.00	1.23
EPI2 [164]	0.71	0.81
EPINET [138]	0.34	0.34
FASTLFNET [62]	0.57	0.58
LF [66]	0.58	0.61
LFOCC [160]	1.70	1.60
OFSY [143]	0.32	0.37
RM3DE [108]	0.73	0.72
RPRF [61]	0.66	0.64
SCGC [139]	1.04	1.09
SPO [183]	0.60	0.71
BSL [6]	0.92	1.01
FSL [8]	0.85	0.95
PSL [5]	0.93	0.89

TABLE 5.4: Evaluation of different algorithms with *Q*25 metric on 4D LF Benchmark [55]. Italics indicate articles written in scope of this dissertation.

	Runtime, s.	
	Median	Average
EPI1 [70]	85.045	88.194
EPI2 [164]	8.789	8.406
EPINET [138]	1.972	1.976
FASTLFNET [62]	0.624	0.625
LF [66]	994.311	1009.756
LFOCC [160]	10614.535	10508.469
OFSY [143]	198.299	200.282
RM3DE [108]	45.149	47.434
RPRF [61]	35.456	34.529
SCGC [139]	2052.190	2056.344
SPO [183]	2111.50	2115.417
BSL [6]	5.149	5.962
FSL [8]	1.766	1.716
PSL [5]	28.811	27.939

TABLE 5.5: Evaluation of different algorithms on their running time on 4D LF Benchmark [55]. Italics indicate articles written in scope of this dissertation.

	$M, \frac{\%}{s.}$	
	Median	Average
EPI1 [70]	0.907	0.858
EPI2 [164]	8.768	9.202
EPINET [138]	48.996	48.112
FASTLFNET [62]	147.051	145.488
LF [66]	0.084	0.083
LFOCC [160]	0.008	0.008
OFSY [143]	0.447	0.439
RM3DE [108]	2.038	1.893
RPRF [61]	2.541	2.606
SCGC [139]	0.044	0.042
SPO [183]	0.043	0.043
BSL [6]	16.817	14.636
FSL [8]	49.875	50.728
PSL [5]	3.068	3.121

TABLE 5.6: Evaluation of different algorithms by the proposed M-metric on 4D LF Benchmark [55]. Italics indicate articles written in scope of this dissertation.

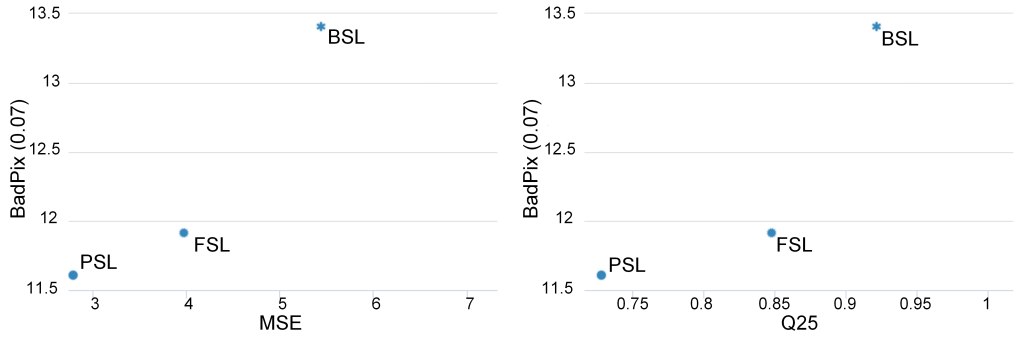


FIGURE 5.13: Comparison of the three algorithms, developed in scope of this dissertation: (BSL [6], FSL [8], PSL [5]) on the median of three metrics (lower is better), generated by [1]

to provide training data. Such approaches can be easily extended to the different configurations of cameras and to be used on various scenes as well, providing not perfect, but reasonable result.

All these algorithms at the time of the publication were implemented on CPU and were forced to be run in the single-threaded mode to better understand their performance in non-parallel configuration. As per analysis of the running time in Table 5.5 we can state that running of BSL and especially FSL configurations is better than in most of the state-of-the-art algorithms. The algorithms runtime was measured on central processing unit (CPU) Intel Xeon E3-1245 V2 @ 3.40 GHz. Implementation was done in C/C++ with the help of OpenCV and Google Ceres libraries.

5.2.5 Discussion

Qualitative results

Big effect on the quality in our opinion was achieved by applying SGM not only to the values of initial, but also to the final disparity map. Practically in can be visible by checking the transition between BSL and FSL on the borders of the object, sharpness of which has increased by changing the SGM application strategy.

However, in the images from both algorithms a so-called "thickening effect" was observed on the object boundaries, which was due to the windowing principle of matching cost construction. By using the per-layer disparity filtering with subsequent holes filling this problem was mostly eliminated. It makes the reconstruction look sharper and closer to the color projection. It can be seen on Fig. 5.15c, that FSL provides different reconstruction on edges, what usually was a spot of ambiguities for the matching algorithm. It happens because pixels around edges were considered as a part of a neighborhood disparity layer due to nature of the matching algorithm, which considers the interpolated color values of pixels among LF views. However, some wrong pixels can still "survive" the filtering, as it can be seen on the left side

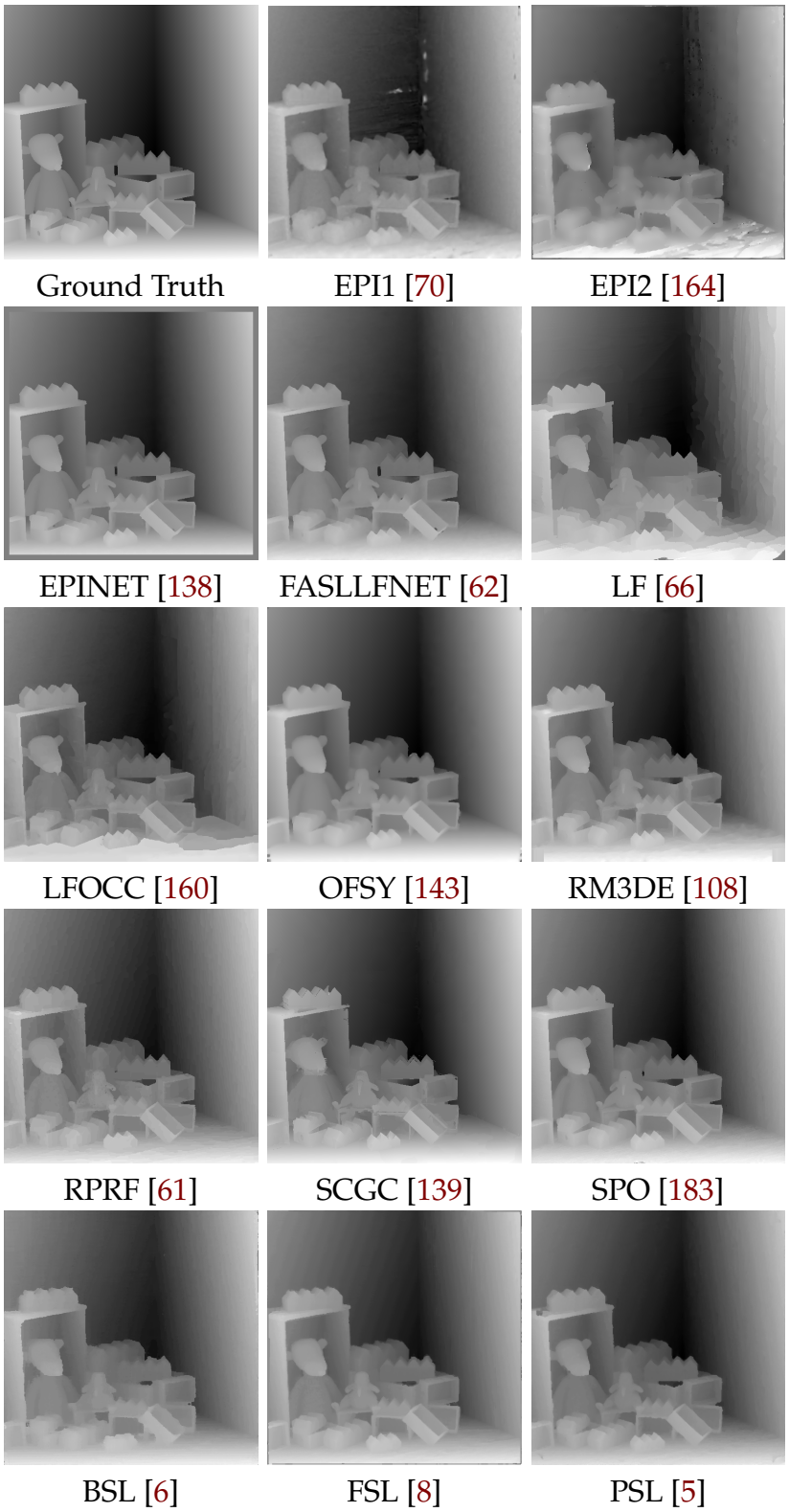


FIGURE 5.14: Qualitative result for "dino" scene from 4D LF Benchmark [55]

(a) (b) (c)

FIGURE 5.15: Effect of the per-layer disparity filtering and holes filling on PSL [5] (a) disparity map from FSL [8], (b) disparity map from PSL [5], (c) difference between two disparity maps

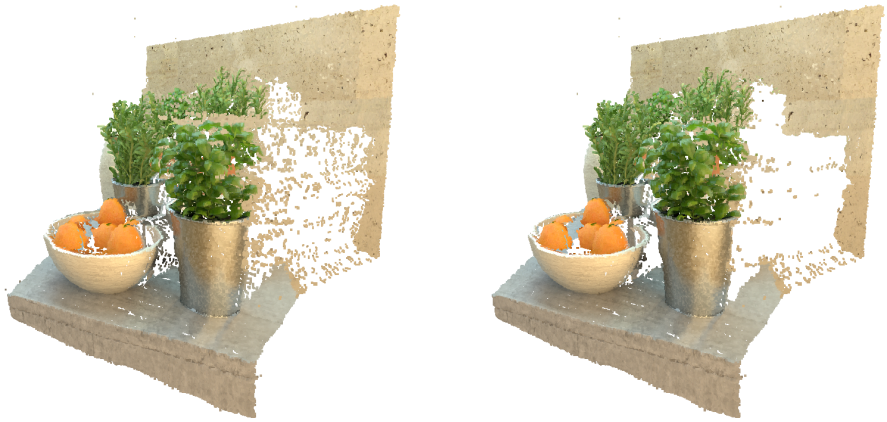
of the statue's head on Fig. 5.15b. Potentially it can be fixed by repeating the per-layer disparity filtering and holes filling several times.

Strong "step" effect on the disparities from BSL and FSL versions of the algorithm is visually noticeable. This aspect was improved in PSL, as can be seen on Fig. 5.15a and Fig. 5.15b. This happens partially due to a change in the subpixel refinement algorithm from the parabola fitting to Symmetric-V. However, most of the smoothness is brought by point cloud refinement step. It is not limited to the discrete matching cost values, unlike the interpolation step. In total this corrects the step effect on disparity values, which was strongly observable in two previous versions of the algorithm.

Running time

Unfortunately, the authors of various algorithms do not always indicate what time has been measured, in particular, what parts of pre- and post-processing data for this algorithm have been included in the reported running time. The methodology by which these measurements were made is also occasionally wrong, for example in the case of using the compute unified device architecture (CUDA) framework with other neural network frameworks an important point for measuring running time is to call the synchronization function, which is not always done, according to our analysis of open source code, and can also lead to incorrect (mostly underestimated) measurements of running time.

The deep learning-based approaches EPINET [138] and FASTLFNET [62] can be considered as top-of-the-line, providing good results in terms of depth quality together with the satisfying running time, which is especially noticeable by the results of proposed *M*-metric. However, their algorithm is performed on the high-end GPU. The amount of required computing power and memory consumption for such algorithms exceeds the capabilities of current embedded hardware solutions by an order of magnitude. Such heavy computational resources are not required by our approach, which utilizes a CPU without specifically employed thread parallelism.



(a) (b)

FIGURE 5.16: Effect of disparity boundaries on the point cloud: (a) point cloud without borders, (b) point cloud with borders

Matching cost bordering effect

Borders from the initial disparity map help to reduce the number of sampled hypotheses by further processing in a range from 50% (real-world scenes) up to 97% (synthetic scenes). Runtime of the correspondence search in the LF space is affected proportionally, since usage of borders significantly reduces the number of sampled hypotheses.

Due to the change of domain from disparity maps to point cloud in PSL a new advantage of using the boundaries was observed. Previously only the running time-related changes were noted, however it turns out that image initialization also prevents the creation of noise pixels on the areas of big disparity values transition, like it can be observed on Fig. 5.16.

In some scenes, SGM results were eliminated for a large number of pixels, and line fitting without bordering information for these pixels affected the runtime (scenes "bicycle", "herbs", "boxes" and "sideboard" from 4D light field benchmark (4DLFB)). This occurs because of the amount of fine structures in the scene. Application of a smaller window for census transform seems to be a solution in terms of accuracy of the borders for the fine structures. However, it reduces the quality of the whole image, hence we decided not to use it.

Although bordering of disparity values with the initial map helps to reduce disparity mismatching noise, some of the wrongly calculated pixels still can "survive" this filtering, which is visible in images on Fig. 5.17. These mistakes appear either in the areas marked previously as non-consistent or on the object edges.

FSL configuration fails with disparity estimation on image boundaries for scenes with a relatively large distance between LF views. The explanation of this problem is related to our selection of the central LF image as a reference view. In this case search for a matching pixel from image boundary fails since there is no match in most of the images and therefore our algorithm can not aggregate enough depth score for the correct value.

	BadPix	*MSE*	*Q*25
With	9.89	3.57	0.74
Without	10.23	5.72	0.72

TABLE 5.7: Average results on "training" subset of 4DLFB [1] for the configuration with and without per-layer disparity filtering and holes filling

A solution for this problem was proposed by Kim *et al.* [75], where the position of reference images changes over time and cost aggregation is performed from the new position. It was further incorporated in PSL configuration as generation of point cloud from the difference reference view with further point clouds fusion.

Effect of filtering techniques

Table 5.7 shows the quantitative difference of algorithm configurations with and without per-layer disparity filtering and holes filling, performed on a subset of images from the benchmark.

Additionally, it can be observed that filtering techniques not only affect the boundaries of the images, but also improve the accuracy of the algorithm on *MSE* metric.

Point cloud refinement and nonlinear optimization

Non-linear optimization step requires a good initialization. Also, such optimization on a full disparity range can unfortunately create additional false estimations. For that, the searching range is limited to 1.5 pixels around the initial value from final disparity map.

One way of improvement of this step, as well as general generation of bordering information, is related to utilization of matching cost confidence measurements. Different thresholds can be used based on the accuracy for the specific pixel. An overview of methods for that is presented in [59].

Unlike previous approaches, running time of PSL was significantly higher. Main reason for that is the non-linear optimization. It can be reduced by using the architectures which supports parallel estimations, like GPU, for such optimizations.

Matching cost selection for real-world applications

The main aspects in the problem of choosing matching cost for real systems is the need to strike a balance between the quality of the result, the computational simplicity of the computation, and the resource constraints of the system. In practical applications, the choice of matching cost is crucial as it directly affects the accuracy of the results.

We experimented with different matching cost configurations to calculate initial and final disparity maps. In the case of calculating the initial disparity map for both synthetic and real images, the obvious choice based on the results of these experiments for us was the matching cost based on census-transformed images. This decision is based on the quality of the initial map generation result. In the presence of a small number of images for the matching cost set, the data collected with $L2$ are less informative in comparison to the census-based matching cost. More specifically, in the case of $L2$, the potentially arising ambiguity in the accuracy of determining neighboring or noisy regions is intractable. In the case of census transform this is compensated by the presence of a window for the actual collection of the data fro this transform.

The choice of matching function for the case of estimating the final disparity map from all LF views is less obvious. Data from multiple LF views is usually enough to dial in a sufficiently representative matching cost and to smooth out all the problems associated with ambiguity in the calculations. However, in the case of real-world capturings, the decisive factor is the resistance to noise in the images, which in one way or another will be present in them. In the case of census transform as a measure of comparison, in addition to its window-based essence, a crucial aspect is its nonparametric basis, since it is not the pixels themselves that are compared, but the ratios of these pixels. Thus the presence of noise in the images is mitigated by this.

Fig. 5.17 shows the comparison of the result of $L2$ and Hamming-based matching cost on the real-world images. First three scenes are provided by EPFL dataset [127], edited and uploaded by C.-T. Huang for the publication [61]. LFs consist of 3×3 RGB image with resolution 541×376. The rest of the images are generated using 3D LFs provided by Middlebury 2006 dataset [54, 133]. Each scene is represented by 7 images in a row with a large baseline between them. Original resolution is 1240-1396×1110 pixels, we used half-sized images for our experiments.

5.3 Conclusion

In this chapter the development of the efficient LF depth estimation algorithm was explained. We show the main techniques to reduce the computational load to compute the depth map with minimal loss of quality. We examined different methods of matching cost aggregation from the perspective of their applicability not only in synthetic, but also in real-world scenarios. The configuration of the method FSL [8] was in the end optimized and used on the real-world system, published in [7] and discussed in Chapter 6.

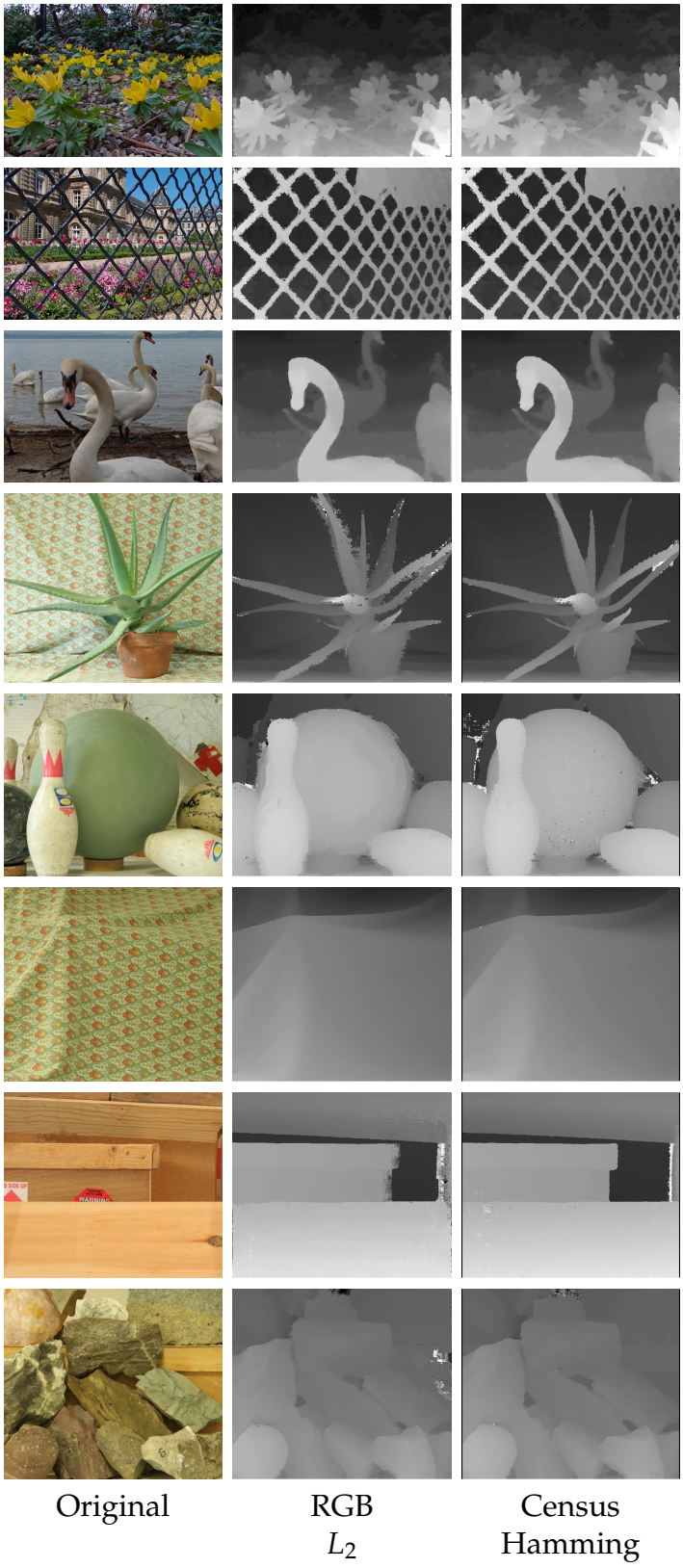


FIGURE 5.17: Qualitative results for real-world scenes with different comparison function for matching cost construction: first three rows – EPFL dataset from [127], the rest – Middlebury dataset [54, 133].

Implementation and applications

The algorithms described in chapters 4 and 5 form the basis for the practical implementation of an entire system for calculating depth maps in real time. This chapter will describe how this system was brought to life, what changes need to be made to the algorithms to run them on the embedded hardware, and what optimizations are key to their performance.

Fig. 6.1 shows the outline of the proposed depth estimation system algorithm. The main parts of it were discussed in Chapters 4 ("Camera calibration", "Camera refinement") and 5 ("Depth Estimation"). We will explain, how the LF capturing device was design and exploited, and how the computational platform is functioning.

6.1 Light Field Camera

The LF capturing device is built on the base of Ximea CB200CG-CM RGB camera, shown on Fig. 2.17. The camera by itself is based on CMOSIS CMV20000 image sensor with 20 megapixel resolution. A 4×4 single lenses array is placed in the front of the image sensor. From 5120×3840 pixels of original images 16 images of resolution 960×960 can be extracted by cropping. Taking into account the circular region formed by the lens within the image, the maximum effective resolution of a LF view was determined as 704×704 pixels. The baseline between two lenses is equal to 0,006 m. We define the working distance of this LF camera as 0,5 – 2,0 m as the compromise of the acceptable number of verified disparity hypotheses for the real-time processing (lower bound) and the distance, on which physical error of estimated depth is still acceptable (upper bound).

As stated in Section 2.3.5, such a camera assembly approach is easier to implement and more robust to manufacture compared to micro-lens approaches. Our camera setup is somewhat similar to the multi-camera LF capturing devices presented in [131]; however, since all of the LF views are captured by a single image sensor, it can be considered as more resilient technology, because cameras synchronization and compensation of single sensors deviations do not need to be performed. The connection of the camera to the computational platform is provided via PCIe interface.

On the driver level the camera provides access to different parameters, which can be adjusted using the camera's API during it's operation. The first important parameter is camera exposure time, also known as shutter speed,

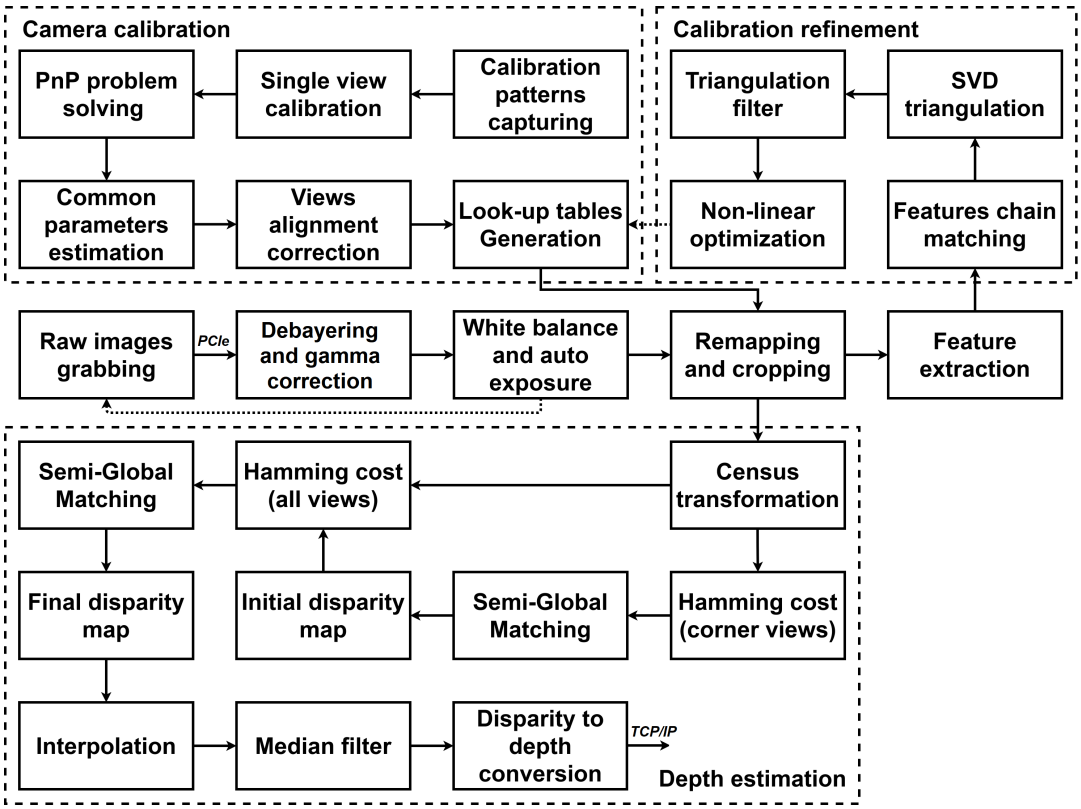


FIGURE 6.1: Overview of the system implementation

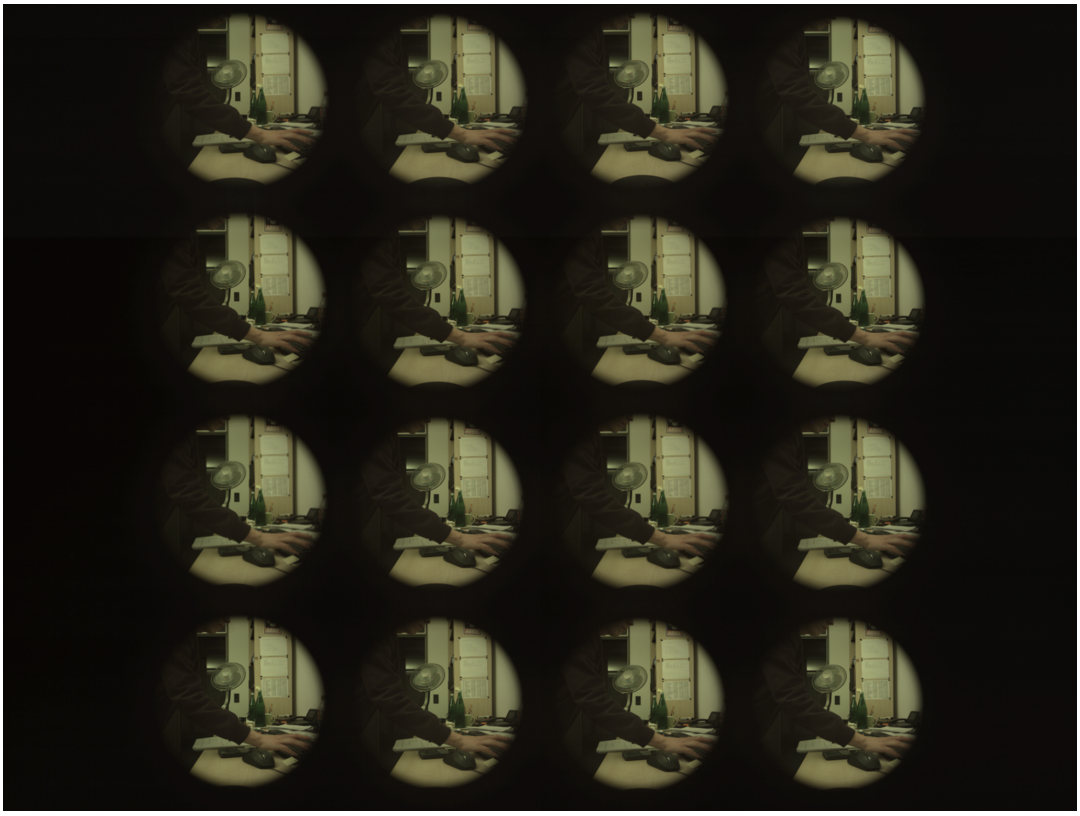
which is the length of time that the camera's shutter remains open, allowing light to enter the camera and reach the image sensor, measured usually in fractions of seconds. A longer exposure time allows more light to enter the camera, resulting in a brighter image. This can be useful in low-light situations or when trying to capture motion blur in a moving subject. However, it can also result in blur from camera shake or subject movement, and apparently affect the running time of the whole system, if this time is too big. Conversely, a shorter exposure time allows less light to enter the camera, resulting in a darker image. This may be desirable when shooting in bright sunlight or trying to freeze motion in a fast-moving subject.

Images obtained by the camera are requiring some pre-processing for their usage. Some of them are needed for the basic conversion of the raw image to perceptible one, while others provide suitability of the system to real-world work conditions. These methods are discussed in the next section. The example of the post-processed frames together with the principle of view cropping are visualized on Fig. 6.2.

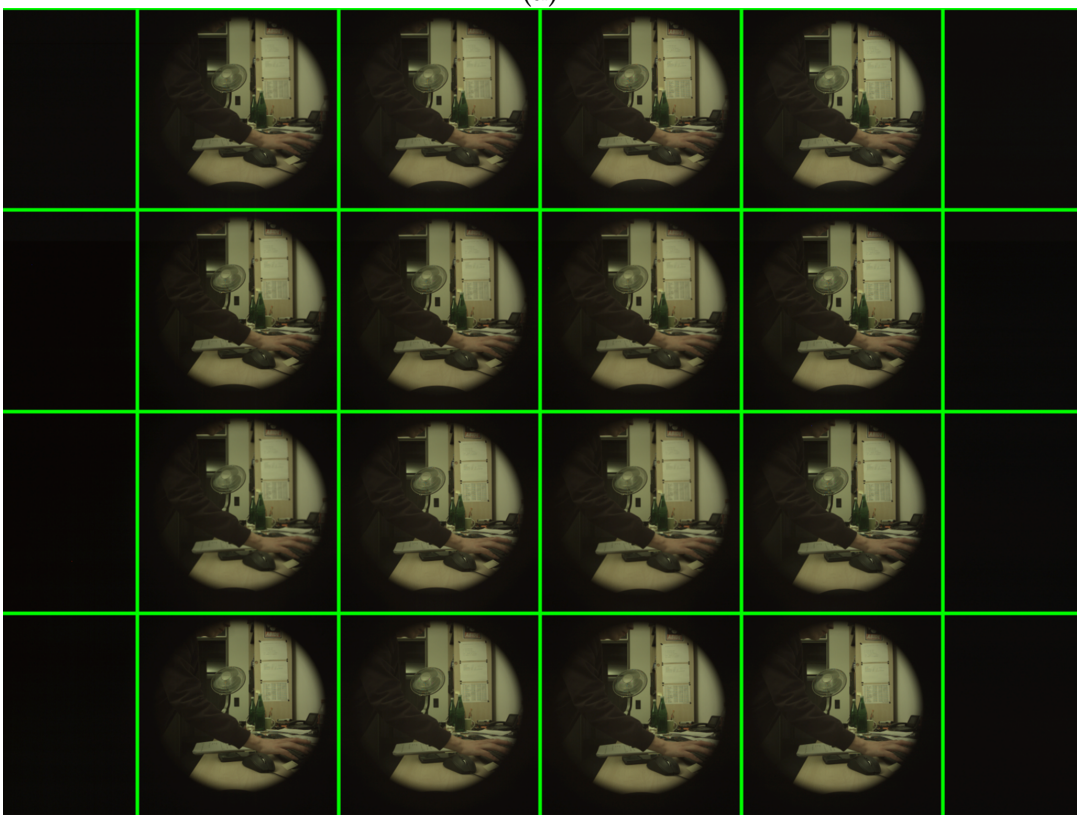
6.1.1 Images pre-processing

Debayering

A color filter array, which captures the specific light waves, was patented by Bayer [11] and now is widely used for the image sensors in modern cameras. The Bayer pattern consists of a grid of color filters that are arranged



(a)



(b)

FIGURE 6.2: Pre-processed frame (a) and the principles of views cropping (b)

in a specific pattern over the sensor pixels. The pattern typically consists of alternating red, green, and blue filters, with twice as many green filters as red or blue. This is because the human eye is most sensitive to green light, and by using more green filters, the camera can capture more detail and color accuracy in the resulting image. When an image is captured using the Bayer pattern, each pixel records only one color component - either red, green, or blue. This means that the resulting image is in a raw format, and needs to be processed to reconstruct the full color image.

The process of reconstructing a full color image from a Bayer pattern image is known as demosaicing or debayering [76]. This involves using algorithms to interpolate the missing color components from the neighboring pixels, based on the color filters in the Bayer pattern. This process is quite time-consuming, since it must be applied to the entire image and involves interpolation of values. Fortunately, due to the fact that usually groups of pixels taken with this pattern are independent of each other, it is possible to process these groups in parallel.

Gamma correction

Gamma correction is non-linear transformation, which serves for the adjustments of brightness and contrast of the images [99]. It tries to compensate for the differences between the way that human eyes perceive light and the way that digital sensors capture light. The concept of gamma correction is based on the fact that the human eye does not perceive brightness in a linear fashion, as it is provided by the sensor. Instead, our eyes perceive changes in brightness logarithmically, meaning that we are more sensitive to changes in darker parts of an image than in brighter parts. Gamma correction involves applying a power law function [109] to the pixel values of an image to adjust the brightness and contrast. This function is typically represented by a gamma value, which controls the amount of adjustment applied to the pixel values. A gamma value greater than 1 will darken the image, while a gamma value less than 1 will brighten the image.

Fig. 6.3 visualizes these two steps of image data pre-processing.

White balance

White balance is one of the methods for the global color balancing, or intensities adjustment, in the image. It is done to ensure that the colors in an image appear natural and accurate, regardless of the lighting conditions under which the image was captured. The basic principle behind white balance is that different light sources emit different color temperatures of light. For example, indoor lighting typically emits warmer, yellowish light, while daylight tends to emit cooler, bluish light. White balance is needed to bring the images to the unified color view regardless the temperature of light source. Also, it helps to correct the cases when specific channels of the image sensor have a larger multiplier, as it was in the case of our LF camera and can be visible on the Fig. 6.2, where frames look somewhat yellowish.

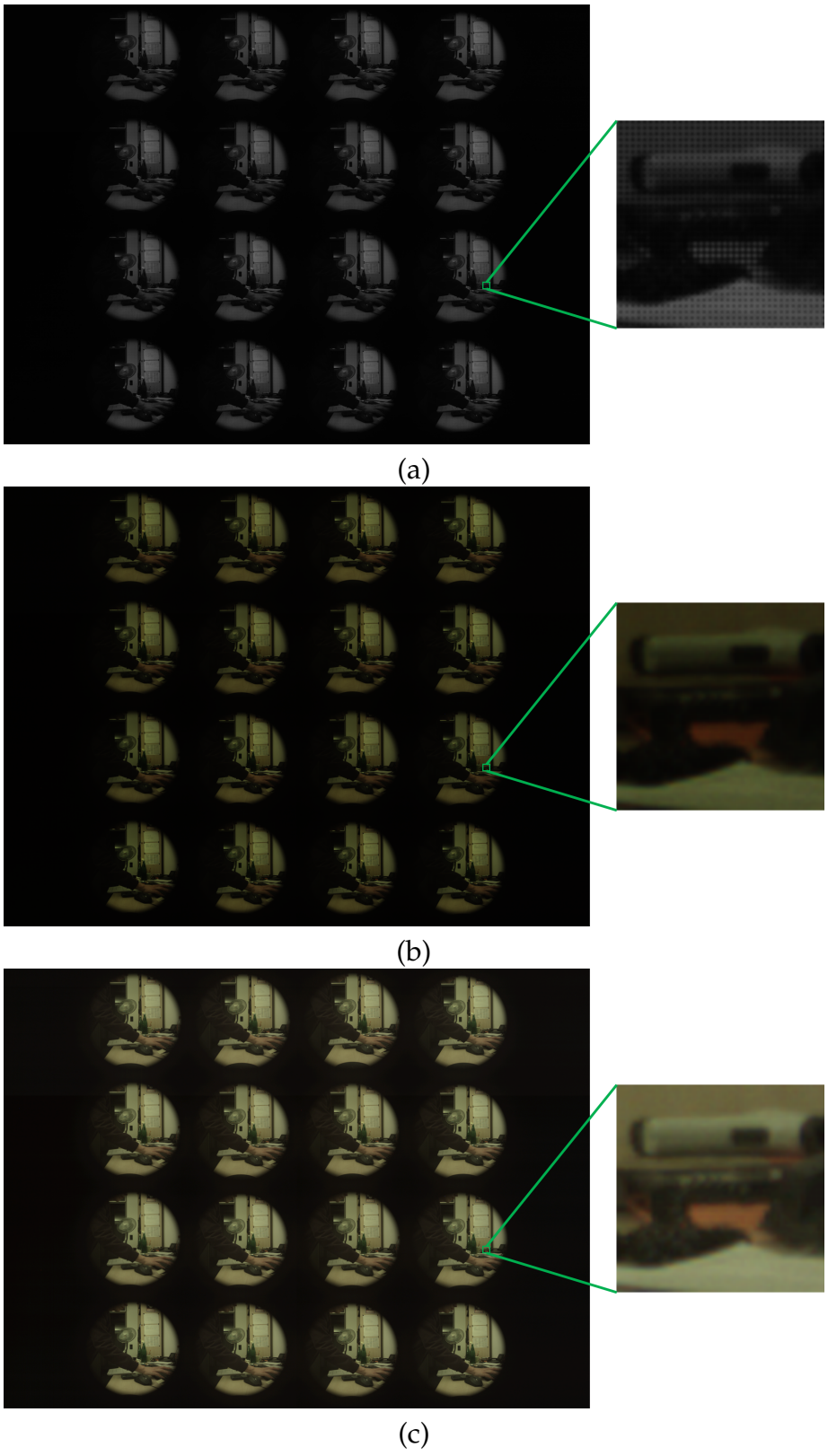


FIGURE 6.3: Steps of pre-processing: (a) raw (bayered) image, (b) debayered image, (c) gamma-corrected image

FIGURE 6.4: Demonstration of auto white balance

While many white balance algorithms may work with specific presets, we use completely automatic calculations. The idea of white balance is based on the gray world assumption [24], which assumes a single grayscale value as a result of averaging the image. Having a reference LF view in RGB, our method first finds the mean value of every channel separately, does the conversion of this view to grayscale and finds the mean value of it. Coefficient for every channel is then defined as the ratio of the average gray value and the average value of this channel. Estimation of coefficients for white balance through the whole image is quite a computationally-intensive operation due to the involvement of all pixels. During the experiments, we found out that taking a grid of pixels from the image to estimate the white balance parameters leads to the similar result compared to taking the whole image, but takes proportionally less time. The values of each of the channels are multiplied by their respective coefficients for the balance.

Fig. 6.4 shows how the white balance algorithm changes the image. The photo shows a camera installed in a test room with a color temperature-controlled light source and an image from the camera subjected to white balance. It can be observed that the algorithm, despite the yellow light, helps to display the image in a more unified and light temperature-independent manner.

Auto-exposure

During the camera exploitation in various conditions it is not possible to manually adjust the shutter speed according to the incoming light intensity. Moreover, the selection of the optimal exposure value might be a non-trivial task for the individual. With that in mind we designed the auto-exposure method, which can process the images and estimate the proper exposure time within a small number of captured frames. The auto-exposure algorithm is used to optimize the shutter speed so that the image sensor can get enough

light to capture the image without over- or under-exposure. It changes the shutter speed accordingly to the total illumination of the scene.

In our pipeline, automatic calculation of exposure occurs together with the calculation of coefficients for white balance. Having the exposure value for previous frame, its modification is defined as the ratio between the reference gray value and average gray value of the current frame.

For both white balance and auto-exposure the grid spacing of 10 pixels for sampling was empirically determined to be optimal. These algorithm work in a range of several captured frames, which provides fast adjustment of the image and camera parameters, which is especially important for the cases of rapid light conditions change (*e.g.* when the car with mounted camera drives out of the parking lot).

6.1.2 Images cropping and remapping

After applying the pre-processing adjustments to the images they need to the undistorted and remapped in the way as they were taken by cameras with same intrinsic parameters. As was stated before it is vital for creating constrains for the further depth estimation.

Based on the position in the original image space, 16 individual views are cropped from this image. Initial intrinsic parameters of every views and their distortion coefficients are determined by the calibration described in detail in Section 4.1. Based on this data, the LUT is generated to store the information about desired pixels positions after remapping. To actualize this information, the auto-refinement pipeline, described in Section 4.2, is executed to adjust the intrinsic values. The used LUT is adjusted accordingly.

6.1.3 Camera and algorithm accuracy estimation

Measuring the accuracy of a camera and algorithm working in the real world is a non-trivial task. It implies the use of well calibrated ground truth sources with a certain level of accuracy. Preferably, such sources should not come from the same domain; in other words, it does not seem right for us to use, for example, one camera-based depth estimation system to determine the accuracy of another, since their result might be flawed as well.

Thus, we utilize data from an external laser measurement device (Bosch Zamo 3) as shown in Figure 6.5. This device has a depth accuracy of $3mm$, which is one order of magnitude higher than our system accuracy and can thus be considered as reliable. For the evaluation, we place our camera in front of a flat wall and align the image sensor parallel to it. To create some invariance to the scene or the texture of the scene and to avoid any influence of the wall texture, different patterns were projected onto the wall. In total, twenty randomly selected patterns with different characteristics were used. Thus, 20 depth images were calculated per distance and their mean depth error was determined. The experiment was repeated for different distances ranging from 50cm to 2m.

FIGURE 6.5: Setup of the distance measurement device on top of the used LF camera

Figure 6.6 presents the depth accuracy with respect to the distance. The error increases quadratically for higher distances, which is mathematically justifiable. For short distances the depth values have an error of below $2cm$, which was comparable to active devices. For higher distances the accuracy is still sufficient for many applications.

6.2 Computational platform

In the process of choosing an appropriate embedded hardware platform for the system design, several essential criteria were taken into consideration, including the capacity for parallel computing, the presence of suitable input and output interfaces (*e.g.* the aforementioned PCIe, Ethernet, and HDMI), and the compatibility with available drivers for the base camera. Upon thorough evaluation of various options, it became evident that the Jetson computing platform from Nvidia met all of these critical requirements.

What shows that Jetson systems are suitable for embedded computations is their relatively low power consumption. It is in the range of 10-30 W, which is comparable to the consumption of a car headlight bulb. The platform supports different power modes, which can be predefined or switched automatically dependently on the actual workload.

Of particular interest to this platform is its architecture, which is used for general purpose computing on graphics processing unit (GPGPU) computations named CUDA. It is a parallel computing platform and programming model developed by Nvidia Corporation for use with their GPUs. At the heart of the CUDA platform is the CUDA programming model, which uses a subset of C programming language and allows to write code that is executed in parallel across multiple threads on the GPU. This is accomplished through the use of CUDA kernels, which are functions that are executed by

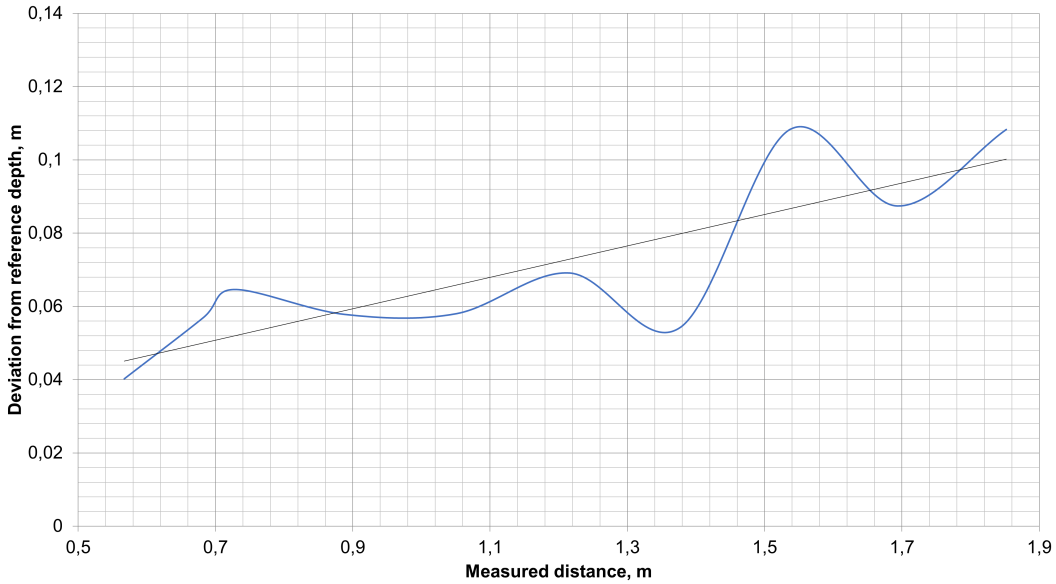


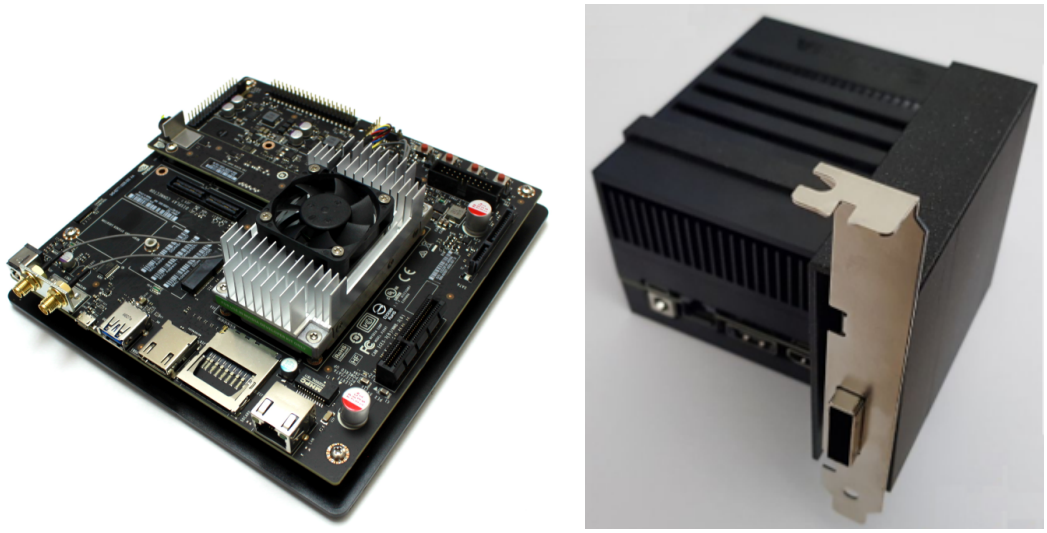
FIGURE 6.6: Dependency of the camera accuracy on the specific distances

multiple threads in parallel on the GPU. The CUDA programming model also includes support for shared memory, which enables efficient communication and synchronization between threads on the GPU.

One of the key benefits of the CUDA platform is its ability to significantly accelerate computationally-intensive tasks that can be paralleled. Another advantage is the native cross-platform code written in CUDA, which allows porting code from one CPU architecture to another (*e.g.* from x86 to ARM) without much hassle. It also includes libraries and tools for optimized GPU-accelerated linear algebra, signal and image processing.

An important feature of CUDA-based devices is the possibility of zero-copy memory access. In this mode, the GPU units have access not only to its own memory, but also to shared random access memory (RAM), which saves time on data transfer and thus reduces latency on image acquisition from the camera. In the platform we have chosen, the memory for GPU and RAM are physically located on the same chip, which saves even more time for zero-copy transfers.

Our system was implemented and tested on two products of this platform. The first platform named TX2 was available as an early prototype of computational hardware of our systems. The device is based on a custom 64-bit ARMv8 CPU and features a Nvidia Pascal GPU with 256 CUDA cores. Its successor, Jetson Xavier, was based on a more advanced architecture Volta with 512 CUDA cores. Both platforms can be considered as complete machines, suitable for work, in a compact form factor. They are shown in Fig. 6.7.



(a) (b)

FIGURE 6.7: Nvidia Jetson platforms: (a) Jetson TX2, (b) Jetson Xavier

6.2.1 Algorithms platform-specific optimizations

The algorithm, used on this hardware, is based on previously described in Chapter 5 FSL configuration. Instead of center view, we selected left bottom corner view as a reference for the depth estimations. This is because the distance between the images, and therefore the difference in coverage in the operating range of the camera, is not so significant.

To implement our algorithms in embedded hardware, we used the OpenVX standard [116]. OpenVX defines high-level abstractions for programming computer vision scenarios, behind which different acceleration engines can be used. The description of the algorithm by OpenVX starts by breaking it down into smaller subtasks, which are wrapped in so-called nodes. These nodes are subsequently combined into an execution graph. The graph is defined once at the start of the application and remains immutable throughout its execution. When a graph is created, the nodes are checked for input and output formats. Low-level calculations are performed inside the nodes. OpenVX can be used to define cross-platform algorithms that are relatively easy to port from one architecture to another, provided that the architecture supports OpenVX.

Using the availability of parallelism, many algorithm details were rethought when transferring from the CPU to the GPU. They mainly concerned the partitioning of easily parallelizable parts of the algorithm into small parts that can be run on separate CUDA cores and whose results can then be assembled into a single structure. An example of such a structure is the matching cost, discussed in Section 5.1.2. Matching cost elements based on census-transformed images can be estimated independently of each other, since they only involve particular pixel values for that.

Procedure to calculate the actual census transform also undergoes some changes on the parallel architecture. In our implementation it is done in a

similar manner to [72], where image is divided to multiple 32×32 blocks, which are consequentially loaded to the shared memory of GPU, in which estimation of census transform based on the pixel surroundings is done faster compared to the naïve approach with pixels comparison in GPU memory. Same shared window principle is used for median filtering. Hamming distance between the bit strings is estimated by applying the *popcnt* operation on the XOR of these two strings. XOR of the two binary strings generates a new binary string that has a 1 in each position where the two strings differ, and a 0 in each position where they are the same. The *popcnt* instruction counts the number of 1's in the XORed binary string. This corresponds to the number of positions where the two strings differ, and thus to the Hamming distance between the two strings [166].

Operations like matching cost construction, left-right consistency check and disparity regression are parallelized trivially, by just doing it separately for every pixel in every considered disparity map. By that, workload can be efficiently distributed among all computational units. An important point for such systems are tests to determine the optimal use of either memory or additional computation. For example, in the case of CPU computing, we used LUT to locate the pixel position in another view of LF, depending on the disparity hypothesis. This was done in order to save time in computing a repetitive operation. However, in the case of the GPU, the computation of this position during the operation of the algorithm was faster than the time to access the memory in which the LUT data was stored.

SGM is an algorithm, which is based on dynamic programming paradigm, in which the results of previous calculations are used for the consecutive ones. Hence, parallelization of such an algorithm is non-trivial and needs to be considered from the position of the possible independent estimations. Luckily, for the case of SGM some estimations can be done in parallel. For example, while traversing in different directions considers different overlapping areas of matching cost, these directions can be traversed in parallel. Also, traversing can be done independently not only per direction, but per disparity hypothesis [72].

6.2.2 Results

Results of the proposed system are shown in Fig. 6.8 The depth result is filtered in a way that it keeps distances in our target range of 0.5–2.0 meters only. The objects are reconstructed with a high level of detail as visible e.g. for the plant hedge. In addition, smaller objects like the pillar are robustly detected, which is important for applications such as the automotive domain. For the flat wall with the homogeneous texture the depth is not very dense, but still depth is sufficiently robust.

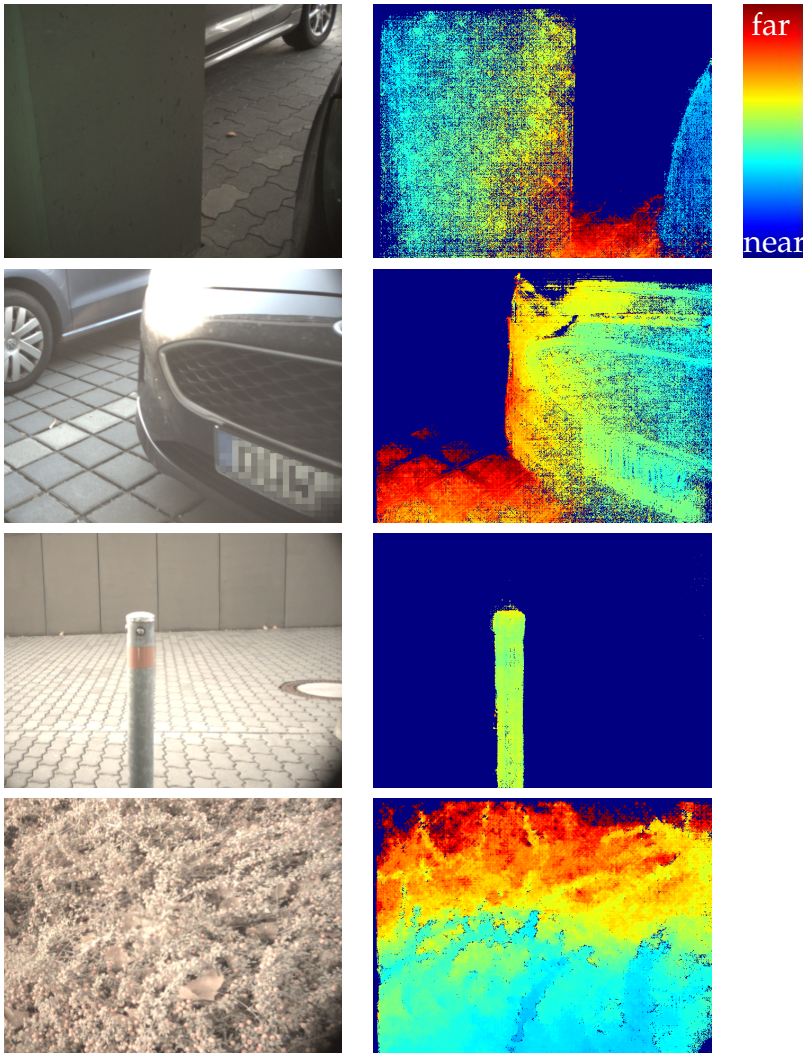


FIGURE 6.8: Qualitative results of the proposed system. The scenes are reconstructed with a high level of detail – even for homogeneous regions (wall), filigree objects (pillar) and crowded objects (plant hedge).

6.2.3 Running time

Being in the time, in which our algorithm can be defined as "real-time" was the crucial part during this work. We measured the running time of the algorithm on Jetson Xavier and compared them with the ordinary PC (CPU: Intel Xeon E5-1620 v3, GPU: Nvidia GTX 1080 Ti). The running times are given in Table 6.1. With our algorithm we can achieve the real-time performance on the different GPU-based architectures, including the embedded one.

6.2.4 Communication interface

In order to facilitate communications between the system and the application platform, a server-client model is employed, utilizing the widely-used TCP/IP sockets via Ethernet interface as the foundation for data exchange.

TABLE 6.1: System running time for two different computation platforms.

	Jetson Xavier	GTX 1080 Ti
Capturing	30 ms	
Pre-processing, remapping	20 ms	8 ms
Depth (initial)	3 ms	1 ms
Depth (final)	14 ms	5 ms
Sending	10 ms	

This approach offers several key advantages. First, it reduces dependency on the application hardware side, since the Ethernet interface is widely used and supported, and there's no need to write additional system-level drivers to communicate with the devices using this interface. Also, it provides capability for remote data capturing over extended distances, even at distances exceeding 5 meters, which is not possible on the *e.g.* USB devices without active repeaters. This feature greatly enhances the versatility and potential applications of the system in various settings.

To transmit images from our system to the application platform we designed a pipeline which is built upon the server-client model, where "server" is the proposed system together with the relevant hardware, and "client" is a computing device, which is using images from camera and the depth map for its application-related purposes. Interaction between server and client on the transport layer is based on the TCP/IP protocol. Access to the protocol is provided via Berkeley sockets API. On the data link layer, the Ethernet standard is used. After running, the server is waiting for the connection from a client. Once the connection is established, server sends so-called ”acknowledgement” package in order to verify, whether the connected client is compatible with the server transmission protocol (not to be confused with TCP/IP 3-Way Handshake). If this sending is successful and the subsequent answer from client ”suits” the server, it first sends the camera intrinsics, and then the size of both color and depth image. Since these parameters are same for every following package, it is needed to send it only once. After that, server transmits the information in the following order:

1. Number of a package `(int32_t)`
2. A timestamp `(int64_t)`
3. Color image data `(char*)`
4. Depth image data `(char*)`
5. Acknowledgement

This transmission goes in the cycle until either client or server side is interrupted.

During the first connection to the server, the client can send a request for adjusting the depth estimation algorithm settings by selecting different

modes, request for the setting of region-of-interest (ROI) and exposure, framerate and camera gain settings (EFG). Request for modes is provided in a form of 8-bit variable with values, which are explained in the corresponding document. Requests for ROI and EFG are provided in the numerical form within the following ranges:

- start_x: 0 - 960 pixels
- start_y: 0 - 960 pixels
- end_x: 0 - 960 pixels
- end_y: 0 - 960 pixels
- p_exposure: 0 - 33000 ms
- p_framerate: 0 - 32 fps
- p_gain: 0 - 512

The following section explains our choice of the depth transmission format.

6.2.5 Fixed-point depth representation format

The depth information is stored in the memory as a single channel image, in which each pixel is represented by eight-bit unsigned integer with the value range 0 — 255. In the next steps the way of depth image conversion to such a form will be explained.

Metrical depth data is usually stored in floating point numbers. For the decreasing of the data amount needed to be transmitted without significant information loss it was decided to use an integer-based fixed point number format with specified number of fractional bits. This format is presented in technical literature as a "Q-number" and finds its usage in a variety of embedded application. In this case 2 integer bits and 6 fractional bits are used (usually denoted as Q2.6).

Integer part of this number was selected based on the maximum required depth which is stated in the system specification. Number of fractional bits is selected as a balanced one, which allows to save the metric data in eight-bit form and in the same time does not provide a significant information loss, which will be proven later.

Conversion of the float number to the Q-form can be performed by a simple multiplication of the original number by 2^n, where n corresponds to the number of fractional bits in the desired Q-number (in our case $n = 6$), together with subsequent rounding to the nearest integer (denoted as $\lfloor \dots \rceil$. For the floating point number N_f the Q-number N_Q can be found as:

$$N_Q = \lfloor N_f 2^n \rceil. \tag{6.1}$$

The back-conversion is performed as simple as a multiplication of the Q-number, converted to the floating point format N_{Qf} ($20 \rightarrow 20.0$), by 2^{-n}. N_{fQ} stands for back-converted from Q-number floating point number and can be found as:

$$N_{fQ} = N_{Qf} 2^{-n}. \tag{6.2}$$

Representation of the metric values in such a form is related to the information loss up to some percentage, which can be roughly represented as $2^{-(n+1)} * 100\%$. However, for the case of our camera with the specified minimum and maximum distance (0,5 – 2,0 m) such a loss does not affect the final accuracy. The following calculations were made for the numerical verification of this statement. For every $N_f \in R$, $R = \{R_b, R_b + R_s, R_b + 2R_s, \ldots, R_e\}$, the conversion error λ:

$$\lambda = |N_f - N_{fQ}| \ (m), \tag{6.3}$$

where N_{fQ} is computed from the N_Q using Eq. 6.1 and Eq. 6.2, and $|\ldots|$ denotes an absolute value of a number. Having a set of λs for different N_f values we can find a maximum λ, which will correspond to the maximum error value in meters after the conversion. For Q2.6 number format the maximum error value λ_{max} within the range R where $R_b = 1\ m, R_e = 2\ m, R_s = 10^{-3}\ m$ is equal to 0,00775, which is correlated with the previous rough estimation.

In order to determine the geometrical error of the depth estimation we convert two nearby values of the pixel displacement to the metric representation. Assuming that maximal disparity estimation error is $\Delta d = 1\ px$ and taking the closest (to the specified depth) pixel displacement values p and $p' = p - \Delta d$ the maximal depth error on this displacement ΔZ_p is based on Eq. 5.22 and determined as:

$$\Delta Z_p = \frac{fB}{p} - \frac{fB}{p'} \ (m). \tag{6.4}$$

For $p = 32$, as for the value correspond to the closest computation point, this error is equal to $\Delta Z_{32} = 0,01449\ m$. We can see, that this error is about 1,87 times bigger than the conversion error, which proves the sustainability of the chosen conversion coefficients.

6.3 Applications

Our system was tested on two potential applications. The first is related to the industrial system for the assembly assistance. The second application is associated with currently relevant ADAS.

The advantage in case of our system for reconstruction together with taking full-color images is that both images (depth map and RGB) are already in the same coordinate system, which saves time for converting images from one coordinate system to another and allows direct usage of both. Another

interesting advantage of such systems is the case of partial failure of one of the capturing units providing images in LF. In the case of stereo systems when one camera fails (or in a more optimistic scenario, when it is overlapped by some external object) it is impossible to calculate depth maps, whereas with LF one can restore the depth using other cameras to a certain limit, or change the reference view and use not all images, but only part of them, lying on the cross formed by the axes on which this view is.

6.3.1 Application: Industrial assistant systems

Intelligent assistant systems aim to help with the assembly of any mechanism by guidance using image and scene analysis. An example of such a system is presented in [14]. Our tests were done on a similar setup to the one presented in their paper.

Using information from the RGB, one can quite accurately recognize which tool the assembler is operating at the moment. In the case of our system the depth of the scene can be used to understand the location of the assembler's hand in the scene. For example, in the case of assembling a mechanism on an assembly machine with boxes of parts, the depth map allows us to understand which box the assembler is taking the parts from. Also, with the help of depth maps, one can build the so-called forbidden zones, *i.e.* the zones to which access is not permitted or from which taking parts is not allowed.

Fig. 6.9 shows how the RGB and depth map of our system on the test setup look like. The camera was installed on the top of the assembly table, facing the table. Depth for this case provide the sufficient level of details for the listed tasks.

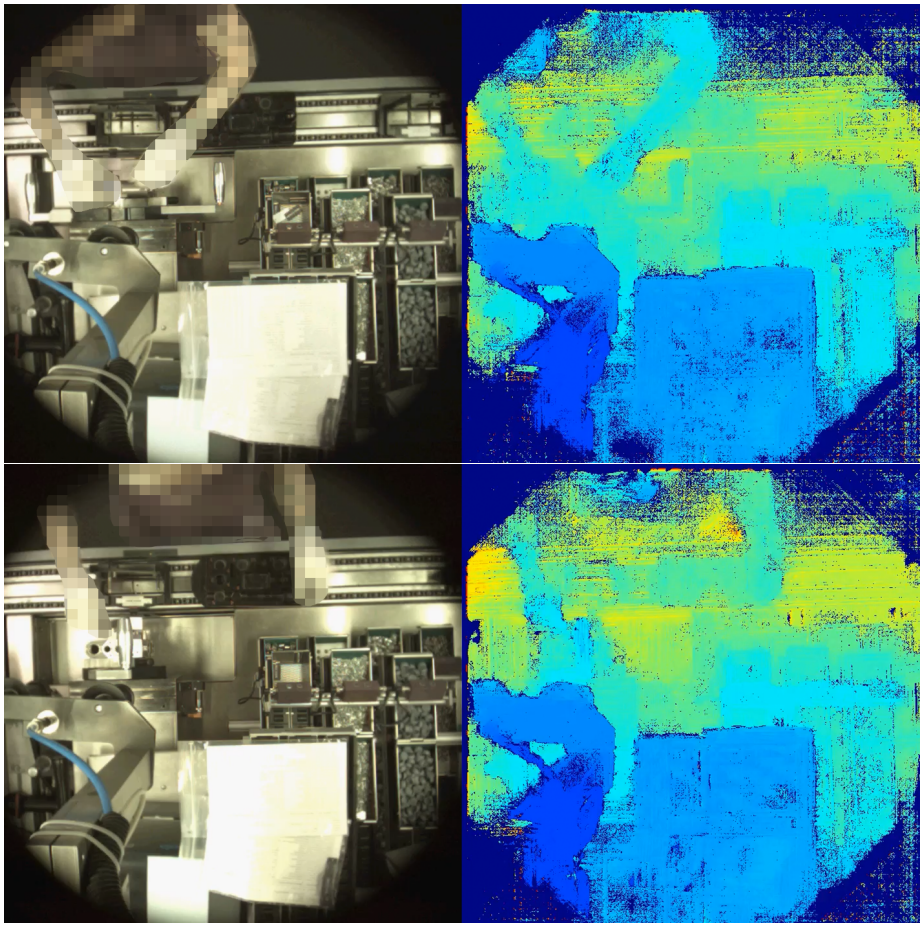


FIGURE 6.9: Example of test recordings for inspection scenario

6.3.2 Application: ADAS obstacles detection

Obstacle detection systems serve as a part of global car ADAS and are designed to help drivers identify and avoid potential collisions with objects on the road. Usually such systems are working based on radars or lidars [173]. However, passive systems are generally cheaper compared to them, while still providing acceptable quality level for such task.

Mapping of RGB and depth within the same coordinate system allows to do more sophisticated analysis of both modalities together. An example of this would be an object detection system that uses either RGB alone [105], or RGB and depth together [122, 162] to pinpoint the type of obstacle.

Fig. 6.10 demonstrates the result of our system. The camera was installed on the backside of the car. One can note good depth quality on small structures, *e.g.* elements of the bicycle.

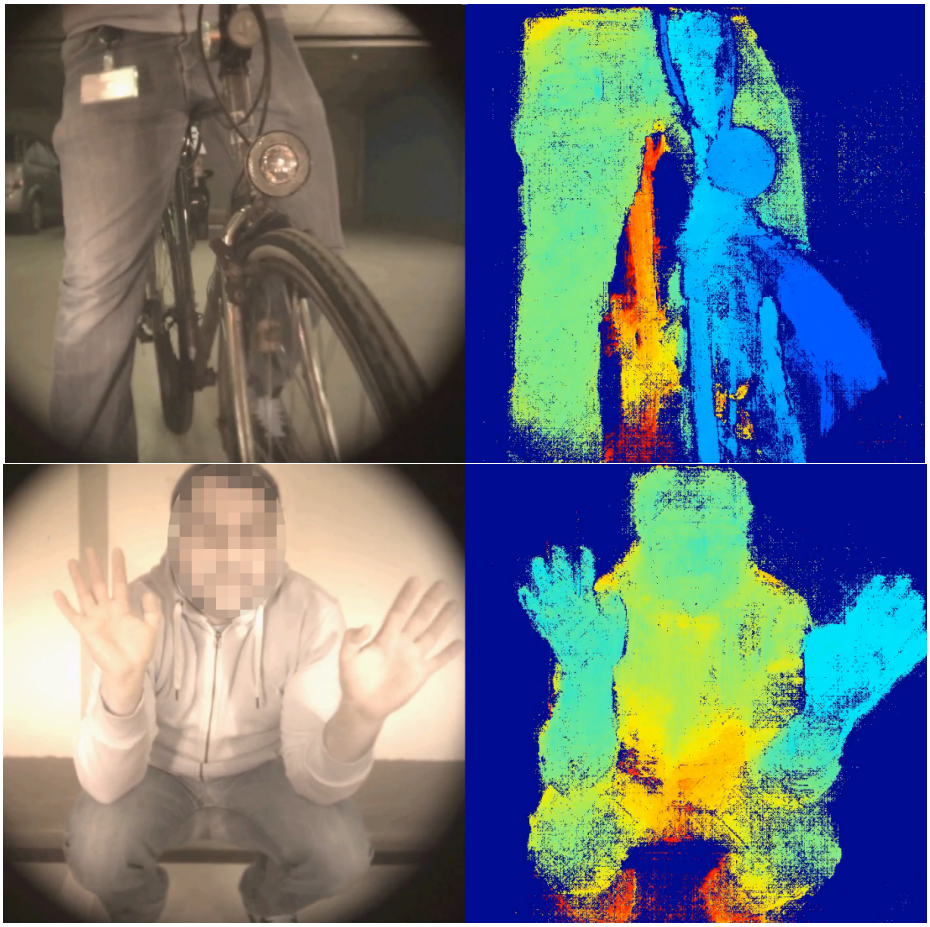


FIGURE 6.10: Example of test recordings for ADAS scenario

6.4 Conclusion

In this chapter the way of embedded system design for LF depth estimation in real-time was presented. The main components were explained. The ways of optimizing the algorithm for the embedded system, as well as the communication and format explanation, were shown. The potential applications of this system were discussed.

The proposed algorithm can work on different cameras without any prior understanding of the data obtained by that camera. However, the main problem of our method is its worse quality compared to deep learning algorithms. In an attempt to find a way to solve the problem of training and tweaking deep learning algorithms for cases where it is not possible to obtain ground truth data, we have developed an unsupervised training method that can be built on top of principally any existing deep learning-based algorithm for finding a disparity map from LFs. Along with the description of this method, in the next chapter we describe potential methods of neural network optimization, which are designed to reduce its size and adapt it for the embedded hardware for the price of some loss of its accuracy.

Deep Learning extension

Classical methods of calculating depth maps, based on geometric properties of the LF, have been used for quite a long time. They allow the computation of depth maps with sufficiently high accuracy and, as shown in the previous chapter, can be adapted to do so with limited hardware resources.

However, in the last few years there has been a clear move away from classical methods towards methods using the deep learning paradigm. Deep learning is a subset of machine learning, where the artificial neural networks with multiple layers are used for the problem solving. Theory of deep learning is well covered in the book of Goodfellow *et al.* [40].

The switch of main computational paradigm from classical to deep-learning based methods is explained by the fact that such methods allow to perform calculations with much higher accuracy in comparison with the classical methods, as the neural networks are probably universal approximators of functions [57], and with their help it is possible to represent a rather complex dependence of the result on the provided data.

In essence, such methods rely on a large amount of input labeled data, in other words, data for which there already exists a ground truth. A large amount of data in simpler cases can be provided by collecting it and then manually or semi-automatically marking it up. However, this approach is hardly feasible for 3D reconstruction tasks. In this case, there are two ways out: either use a ground truth reference source (for example, as was done for Kitti dataset [101], where lidar was used to capture the reference 3D reconstruction) or generate the synthetic data by some rendering engine, which can provide 3D scene reconstruction together with "captured" images.

However, both of these approaches are not without their shortcomings. In the first case there is a question of the accuracy of the reference device, the reduced accuracy of which can lead to incomprehension in the of the neural network estimations. In the case of generating synthetic images we can obtain get the most ground truth; however, such images will not always be similar to what can be shot with a real camera because of their "perfectness".

Another problem is that usually neural networks built on the principles of deep learning require significant computing resources for their operation, in particular, they need a relatively large amount of RAM and architectures that support parallelism on the native level.

As a result of our research on this topic, we will look at existing ways of dealing with these problems. We will consider ways to compress neural networks so that they can be run under hardware constraints. Also, we will

look at ways to adapt neural networks trained on synthetic data to work on real images.

7.1 Neural network compression techniques

Neural network compression techniques aim to reduce the size of neural networks while maintaining or slightly reducing the accuracy of the result produced by them [115]. This is important so that such neural networks can be used in devices with memory and computational resource limitations.

When analyzing what can be reduced in neural networks, two directions seem to be obvious:

- Reducing the size of the neural network, *i.e.* the number of neurons in it.
- Reducing the size of neurons, *i.e.* their bit width.

These techniques are called pruning and quantization and will be discussed below.

7.1.1 Pruning

In the literature, "pruning" means the compression technique that removes either connections between neurons or the neurons by itself [91]. The decision on the removing criteria is based on the necessity or redundancy of the specific connection or neuron. In other words, pruning tries to keep network elements, which impact on the final result is highest.

Reduction of the number of neurons can influence the problem of overfitting the neural network, as in was demonstrated in the early works on neural networks pruning [126]. There, a brute-force pruning method was formulated as setting every weight to zero and evaluating the error of the resulting network. Due to the computational complexity of such an approach, it is not currently used, but other alternative methods are described below.

In modern neural networks there are several principles of categorization of pruning methods [91]. As mentioned earlier, methods can be classified according to which elements of the neural network are removed: connections or neurons. a more detailed classification considers the subdivision into the static methods, which perform the pruning before the inference of the network, and dynamic methods, which are doing it during the inference.

In general, static methods rely on selecting elements for pruning, removing them and then optionally adjusting the result [43]. Criteria for selecting elements for subsequent pruning are magnitude, *i.e.* the numerical value of a certain element, which is either equal to zero or below a certain threshold; or a penalty, which is added to the loss function to detect unimportant network elements.

Dynamic pruning methods, in turn, do not change the structure of the network post-factum, but try to understand during inference which elements of the neural network contribute the least to the result. It helps to reconfigure

the neural network more efficiently for a specific use case on-the-fly. However, compared to the static methods it potentially creates additional computational overhead.

Thus, it can be concluded that static pruning methods are more suitable for use in embedded systems. They directly affect the size of the network and do not create additional computations at runtime, which directly reflects the needs caused by the limitations of embedded systems.

An interesting hypothesis was proposed by Frankle and Carbin in [37]. Instead of removing parts of network, their method tries to find subnetworks (named "winning tickets") inside the bigger networks, which can reach the accuracy similar to the original network after the training. This hypothesis was supported by the training and evaluation of "winning tickets" from popular architectures, resulting in 10-20% size reduction while maintaining similar accuracy.

7.1.2 Quantization

Nowadays, neural network parameters are stored in 32-bit floating-point format. This is done to better preserve the accuracy of the network weights. However, such representation for some cases can be considered cumbersome and overdetermined. Yet another approach to reduce the size of the network while maintaining either the original or pruned architecture is to reduce the bit width of the neural network elements, called quantization.

The term "quantization" comes from digital signal processing, where it means the conversion of an analog signal into a digital one by sampling the values of this signal at certain time intervals [141]. In the context of neural networks, the term takes on the meaning of an operation to represent neural network elements using a smaller bit width compared to the original representation [115].

There are different forms of quantization. In general, they can be divided to post-training quantization and quantization-aware training. First group of methods perform the bit-width redundancy after the training was done on the full bit resolution. Typically, these methods involve converting the weights and activations of the neural network from higher-precision data types, such as 32-bit floating-point numbers, to lower-precision data types, such as 8-bit integers or fixed-point numbers. It can be applied to different elements of the network, such as the weights, activations, or to both.

The second group of methods involves training of the neural network with low-precision data types from the beginning. During training, the weights and activations of the network are represented with lower-precision data types, allowing the network to learn to operate with low-precision data types from the start. Compared to the first group, it can result in better performance and accuracy compared, since the network is optimized to operate with low-precision data types.

By the principles of used arithmetical operations methods can be divided into the following [91]:

- Lower numerical precision methods involve representing the elements of the network using lower numerical precision data types, such as fixed-point or integer. This results in smaller memory requirements and faster computation times, while keeping multiplications as the main arithmetic operations.
- Logarithmic quantization converts the network elements to power-of-two numbers. This allows for the use of bit shifts instead of multiplication, resulting in even faster computation times.
- Plus-minus quantization, where binary or ternary representation of network elements is used. This allows for the replacement of multiplications with additions or bitwise operations [125], resulting in significant reductions in computational complexity.

In general, depending on the requirements coming from the hardware, it is justified to use a particular method of quantization. In the case of depth estimation systems, since the accuracy of the algorithms is critical, the choice of quantization method should start from reducing the bit width of the network elements, and then, if accuracy can still be maintained at an acceptable level, verified on more extreme size reduction techniques.

7.2 Neural network real-world adaptation technique

A majority of the methods require big amounts of data for its training and, more important, the available ground truth data. While for synthetic images the ground truth generation does not pose a problem, for the real-world dataset it might not be available or its capturing may be associated with certain difficulties.

Lack of ground truth data can be partially compensated by using so-called unsupervised training methods. In this type of methods, the dependencies of the input and output data are formed indirectly based on the network output.

Another problem which may arise with real-world LF data is the quality of captured images. Images captured by real cameras may suffer from noise caused by the principles of camera sensor operation. In some cases, individual camera lenses may have slightly different focus, which also affects the result of reconstruction. In the case of constructing LF cameras from several isolated cameras, a potential problem is the slightly different color transmission of each camera.

Throughout a period of time, window-based matching functions have been used in classic depth estimation algorithms to deal with these drawbacks. One of the best examples of such functions are the census and rank transforms, which were discussed in Section 5.1.1. Being non-parametric, these transforms utilize the relative ordering of local intensity values rather than the intensity values themselves.

For the deep learning algorithms, the main limitation for the use of such transformations is the inability to differentiate them in native form. As an

attempt to use the census transform for the deep learning algorithms, we designed a differentiable approximation of the census transform in the form of a loss function for the unsupervised training of LF depth estimation algorithm. It is based on the comparison of result of the views warping to the common image space based on the resulting disparity map, obtained by reconstruction algorithm, with the reference LF view. The comparison method is based on the differentiable census transform and the approximation of Hamming distance estimation, mimicking the original behavior of these functions in the classical methods. Such an approach can work by being embedded to any LF depth estimation algorithm, which takes a LF image as an input and provides the disparity map as an output.

7.2.1 Images reprojection

As mentioned earlier, algorithms that rely on the unsupervised pipeline do not require ground truth data. Instead, they aim to extract valuable information directly from the input and learn the underlying dependencies from it. If we are considering the computer vision domain within 3D reconstruction application, information about the correctness of the disparity map estimation from many images can be evaluated by re-projecting these images into the coordinate space of other images using the disparity map and then comparing the coincidence of the re-projected image with the one originally in its place.

A distinctive feature of the LFs in computer vision, arising from two-plane parameterization, discussed in Section 2.3.2, is that all its views are located on a grid, *i.e.* they are proportionally equidistant from each other and the relationship between the views can be described straightforwardly using the information about the location of the views without taking into account any data about their physical location (of course, in the case of undistorted and rectified images). Thus, it allows using only the disparity map in reference view coordinates, to reproject it into the coordinate area of other images and make a comparison of images to assess the accuracy of the disparity map estimation. Similar scheme is used in other unsupervised depth estimation algorithms, mentioned in Section 3.2.2.

In general this idea can be expressed as follows:

1. Take a LF reference view with certain angular coordinates $(\hat{s}, \hat{t})$ and a corresponding disparity map d.

2. Select a LF view (s, t) and find a difference between its coordinates and the coordinates of reference view: $\bar{s} = \hat{s} - s$, $\bar{t} = \hat{t} - t$.

3. Create two sets of shifted disparity maps (one per coordinate): $d_y = d\bar{s}$, $d_x = d\bar{t}$.

4. Shift the view (s, t) to the coordinates of the reference one $(\hat{s}, \hat{t})$ using d_x and d_y.

Implementation of the reprojection step is done based on the warping transformation. Set of disparity maps d_s, d_t is used to generate two components of the sampling grid, in which the coordinates of the output images are stored. The warping transformation transfers the source image pixels to the corresponding coordinates in the target image based on this grid using a certain interpolation scheme. Commonly used interpolation schemes include bilinear or bicubic interpolation, which can handle non-integer pixel shifts from the disparity maps.

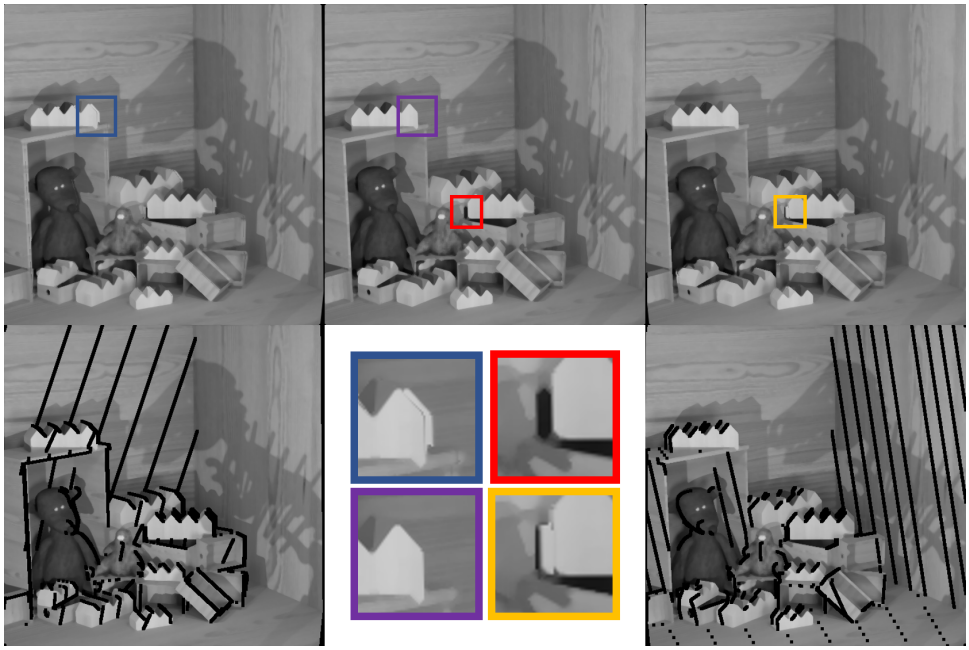


FIGURE 7.1: Demonstration of visual artifacts in occluded areas caused by disparity-based warping and the proposed masking of there artifacts. First row: leftmost view in reference row $(\hat{s}, 0)$ warped to reference view coordinates $(\hat{s}, \hat{t})$, reference view $(\hat{s}, \hat{t})$, rightmost view in reference row $(\hat{s}, |t|)$ warped to reference view coordinates $(\hat{s}, \hat{t})$; second row: filtered leftmost view, sub-images demonstrating the artifacts with color encoding, filtered rightmost view.

Application of warping transformation for images reprojection allows the differentiable processing of the all reprojections steps, which is vital for neural network learning. However, it has a significant drawback, which potentially reduces the quality of the disparity map estimation. As demonstrated on Fig. 7.1, disparity-based warping creates visual artifacts in the occluded areas. During the image comparison phase these artifacts would prevent network to properly converge on these areas, as by using purely warping they can't be eliminated.

In order to address the issue of visual artifacts in occluded areas caused by disparity-based warping, a mask is used to identify valid pixels in the image after applying a shift based on the disparity maps. This mask is derived directly from the disparity map and is generated using the inverted disparity maps $-d_s$ and $-d_t$. Instead of the image, these inverted disparity maps are

applied to the two-dimensional grid, which stores ranges of all the possible image coordinates within the image dimensions. For the leftmost corner of the image, a 5x5 subset of such grids g_x, g_y would look like:

$$g_x^{5x5} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}, \quad g_y^{5x5} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}, \tag{7.1}$$

All the valid unoccluded pixel indices can be obtained by finding a difference between g_x, g_y and corresponding disparity maps d_x, d_y:

$$\bar{g}_x = g_x - d_x, \bar{g}_y = g_y - d_y \tag{7.2}$$

Then, to filter the values these indices are checked for being within the image borders and for being preserved in both grids as $\bar{g}_x \wedge \bar{g}_y$.

Morphological erosion is applied to the resulting mask for removing potentially preserved leftovers of artifacts areas. Resulted mask is used at the loss estimation stage to filter away the values, in which occluded areas were used to estimate the loss.

7.2.2 Differentiable census transform

As was stated before, the main problem with census transform is that it is not differentiable, in other words, it is impossible to obtain its derivative. Consequently, this limits its use in algorithms based on deep learning. To overcome this, we propose the close approximation of census transform based on the differentiable operations.

The main problem for creating a differentiated census transform is the operation of comparing the values of the two pixels. A reasonable replacement is finding the difference between two images, the original and the shifted one. The image shift is done via warping, mentioned in Section 7.2.1. In this case, the shift is applied to the entire image.

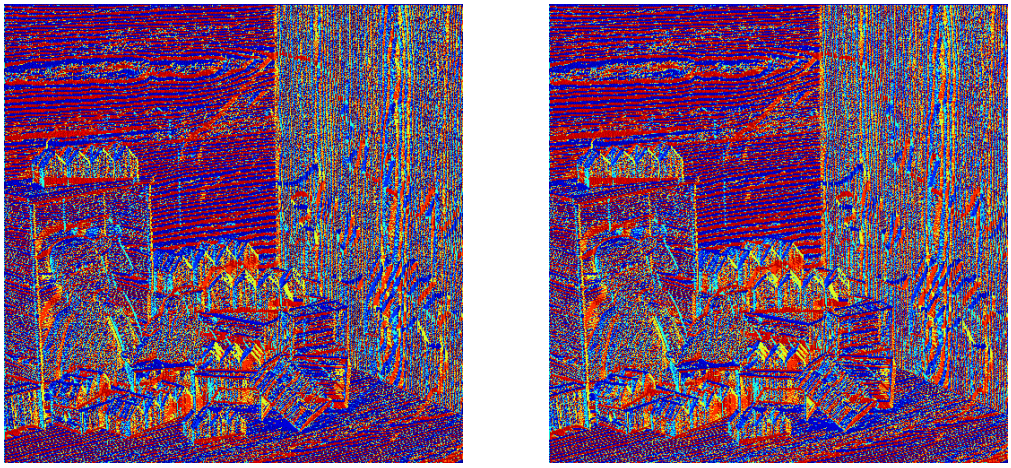
In order for the difference values to form a structure similar to the census transform, the results of the difference calculation are used as an argument for a hyperbolic tangent function with a certain constant C:

$$I_C(u, v, c) = tanh(C(I(u, v) - I(u + i, v + j)))0.5 + 0.5 \tag{7.3}$$

where the channel ordinal number c is defined by the position of i, j in previously defined M. It results to the following:

$$\xi(v_1, v_2) = \begin{cases} 0, & v_1 < v_2 \\ 0.5, & v_1 = v_2 \\ 1, & v_1 > v_2 \end{cases} \tag{7.4}$$

As a result, we obtain N-channel census image, every channel of which cor-



(a) (b)

FIGURE 7.2: Colored visualization of two census-transformed images: (a) original census transform, (b) proposed census transform

respond to the particular bit position from original census transform definition. Fig. 7.2 shows, how the results of original and proposed census transform look like.

The Hamming distance function from Eq. 5.8 in this case can be replaced by the sum of the absolute differences between the two images whose similarity is measured:

$$HD(x_i, x_j) = \sum_{k=1}^{n} |x_{ik} - x_{jk}|, \tag{7.5}$$

7.2.3 Loss function design

The described algorithms for image reprojection, census transform, and Hamming distance estimation are integrated into a loss function that is independent of the neural network model and can be used with any model that operates on LF images.

The loss function takes the LF as input, along with a computed disparity map and information about the reference view position. The algorithm then uses reprojection to warp the images, computes the census transforms for the warped images, and estimates the Hamming distance using an unoccluded filter mask. The result is an estimation of the loss, which can be further back-propagated to the neural network model to adjust its weights.

7.2.4 Experiments

To train and verify the feasibility of this method on the real-world data two datasets were used. First dataset was captured by LF camera, used in the system presented in Chapter 6. It consists of 310 consecutively captured LF of angular resolution 4×4, undistorted and rectified by the method from Section

	Supervised	Unsupervised L1	Unsupervised Census
boxes	24.55	35.04	34.73
cotton	3.08	10.85	7.41
dino	7.06	18.56	17.19
sideboard	14.36	23.83	23.91

TABLE 7.1: Results of supervised training, unsupervised training with $L1$-loss and unsupervised training with Census loss of model from [163], evaluated on subset of 4DLFB [55] by a BadPix metric

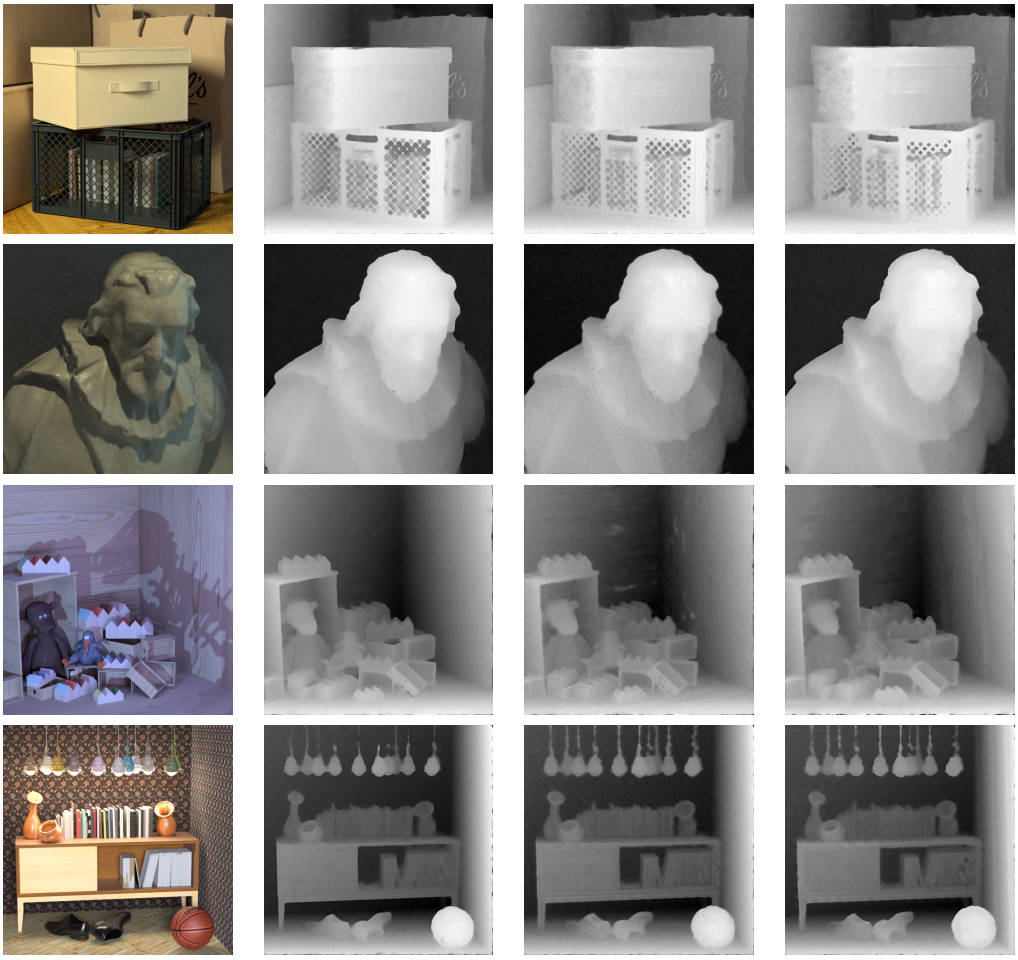
4.1, resulting in 16 RGB images of spatial resolution 512×512 pixels, recorded by moving the camera on the trolley on the parking lot. Compared to the other dataset, two specific challenges of this dataset are the relatively high level of image noise and the slightly different focus in every LF view, which derives from the non-firmly fixes lenses. This brings additional issues for the depth estimation tasks, with which census transform supposed to deal better compared to direct image values comparison.

Second dataset is a subset of LF dataset from Stanford [2]. Original dataset consists of 12 scenes, acquired by a DSLR camera placed on the self-made gantry, resulting in 17×17 set of images per scene. To properly match with the angular resolution and the disparity range of the previously described dataset, from original LFs we first extracted and repositioned 9×9 images around its central view (inclusively), out of which the resulting 4×4 dataset was extracted.

Evaluation of the unsupervised approaches on real-world LF data can be problematic due to the unavailability of the ground truth data. Because of that we can't perform the quantitative evaluation of the depth estimation results on real-world images. Therefore, we perform the evaluation on synthetic dataset with available ground truth for obtaining the statistical measurements. Algorithm's performance is numerically verified on the synthetic data from the previously discussed 4DLFB [55].

For the unsupervised training as a baseline neural network model we used a model, proposed by Wang *et al.* in [163]. Due to the nature of our approach it can be used on top of any depth estimation model, which supports LF; our choice of this model was driven by its code availability within the desired deep learning framework.

The reference model was originally trained on 4×4 synthetic dataset, obtained from 4DLFB. It is done for the comparison of the unsupervised training results with supervised ones. In order to properly compare the usability of the census transform two separated models were trained in the unsupervised manner. One model was trained by directly comparing warped images using $L1$ loss, while the other model was trained by comparing census-transformed images.



(a) (b) (c) (d)

FIGURE 7.3: Demonstration of the evaluation on 4×4 synthetic images from 4DLFB [55], namely "boxes", "cotton", "dino" and "sideboard": (a) reference LF view, (b) results of supervised network from [163], (c) unsupervised network with $L1$ loss, (d) unsupervised network with census loss

Results

Table 7.1 shows the results of the differently trained models on 4DLFB dataset. For training of all models the combination of dataset categories "additional" and "training" was used. While the dataset provides the disparity maps per every LF, they were not used for the unsupervised training part.

We can see that numerically results of unsupervised training are considerably worse compared to the data of supervised training. It confirms that in the general case, with available ground truth supervised methods can achieve higher accuracy than unsupervised methods. The fact that depth estimation methods based on $L1/L2$ distance work better on synthetic images, while census-based method perform better on the real ones, was already observed.

However, the results obtained with unsupervised methods, although not the most accurate, still to a certain extent can accurately convey the 3D data

(a) (b) (c) (d)

FIGURE 7.4: Demonstration of the evaluation on 4x4 real-world images, first row – Stanford LF Dataset [2], second row – our dataset: (a) reference LF view, (b) results of supervised network from [163], (c) unsupervised network with $L1$ loss, (d) unsupervised network with census loss

of the scene. The visualized results of all models are presented in Fig. 7.3. The consistency of our results with the supervised model can be visually confirmed.

The difference between supervised and unsupervised methods is more evident in real-world images. Fig. 7.4 shows the results of differently trained models on the real captured scenes. The difference between supervised and unsupervised methods becomes especially apparent in this case. We can see that although supervised methods more accurately represent sharp edges, in the case of unsupervised training the result looks more dense. Comparing the $L1$-based and census-based unsupervised training, the lower amount of noise on disparity map, which is driven by wrong estimations, is visually apparent.

A serious disadvantage of such a method of computing census transform is the need for large amounts of memory to store it during the training. In the current implementation, every bit of census-transformed images is stored as a separated channel with at least one byte width, which limits its usage as an input for the neural network, as every LF view in such case needs to be transformed. The ways for the proper data compression, which would work in the census transform case and still remain differentiable, need to be investigated.

Currently, the warping occlusions in our case are simply masked away. There are some schemes to take them into consideration, *e.g.* proposed by [65]. Potentially, it can lead to the higher accuracy in occluded areas.

7.3 Conclusion

In this chapter the main methods of neural network optimization for its operation under typical constraints of embedded hardware were discussed. Potentially working configurations suitable for this type of hardware have been suggested.

Also, an approach for the adaptation of LF depth estimation deep learning algorithms using the census transform in an unsupervised manner was shown. The way how to make such a transform differentiable, hence usable for deep learning algorithm, was presented. The method works in a form of a loss function, which can be used with practically any LF depth estimation algorithm in a non-invasive way. Its feasibility was verified on the synthetic and real images, showing its potential for the utilization on the latter.

Conclusion

8.1 Summary

This thesis shows, how the depth estimation system from LFs can be designed to operate in real-time domain with embedded hardware constraints. The environment in which camera operates is defined as challenging, addressing the efficiency-accuracy balance for the depth estimation process, which is rarely covered in the scientific literature.

An analysis of existing algorithm for LF calibration and depth estimation were performed, investigating the existing limitations of these methods to work in the context of defined constraints. Major challenges in this field are the demands of these algorithms for the characteristics of the hardware to perform the computations. The second problem of these algorithms is their running time, which in most cases also does not allow to run them in real time, even on advanced modern hardware. Finally, algorithms for calibrating LF cameras are often difficult in the implementation of components for their functioning.

Trying to cope with these shortcomings, we developed a LF camera calibration algorithm together with calibration refinement method. It allows to conduct simple yet accurate camera calibration, which can be practically done by anyone. The auto-refinement pipeline helps to exploit the camera in the working environment where the recalibration in case of a sudden calibration parameters change is not possible.

Our main contribution is the development of a LF depth estimation algorithm that has low requirements for the hardware on which it is executed. We analyzed existing methods of data aggregation and checked their suitability for estimations of real-world scenes. We proposed to use the information obtained from a limited number of LF views to create boundary information for the subsequent estimations of matching cost from all LF views for the further regression and sub-pixel refinement of the result.

These algorithms were implemented on the real embedded hardware together with the custom 4×4 LF camera. We tested the proposed methods for the optimization of the algorithms to run in the constrained environment and verified their feasibility for it. The ways how the information should be encoded and transmitted for using in various applications was shown.

We analyzed the ways of the perspective deep learning algorithms to be executed on embedded hardware. Methods of reducing the size of neural

networks were considered. We suggested to use the unsupervised loss function to refine the algorithms trained on synthetic data to work on real scenes.

8.2 Potential future work directions

During the development of the algorithms and the end system within the scope of the dissertation, some drawbacks and limitations were faced. We believe that these shortcomings can be addressed by future work, the key points of which we describe below.

8.2.1 Light field calibration

In general, camera calibration is quite a complex task. Even with the proposed procedure, we can see some potential problems, such as the low accuracy of the manufacturing of the calibration pattern, improper positioning of it on the scene, and in general the lack of possibility to make a calibration. Potentially, such problems can be solved by using fully automatic calibration. The presence of many LF views, their predictable location in the grid, and the corresponding chain correspondence matching principle potentially allow for a fairly high level of autocalibration. The now popular deep learning methods can achieve high accuracy for this task.

8.2.2 Light field depth estimation

In general, a potential direction for calculating depth maps with high accuracy is to migrate the algorithm to the principles of deep learning. For more rapid operation, the principles of matching cost bordering can also be used in this paradigm. The use of neural network reduction techniques together with this will allow to bring such algorithms into the real-time processing domain with a negligible loss of accuracy.

During the tests with real-world images, we came up with an idea of adjustment for border threshold λ from Eq. 5.17 dependent on the initial disparity value, so that for small disparities this threshold is higher rather than for higher values, making it somewhat adaptive. Such a strategy can help to determine the more accurate values for the farther objects, where the pixel discontinuity can be a crucial factor for the right depth estimation.

The topic of occlusion-aware processing is of interest, since the specific attention to it allows the reconstruction of partially visible objects of the scene. We also consider census transform as potential operation for the reconstruction tasks due to its stability against image noise and non-parametric nature. For example, census transform might be reformulated to the adaptive case, using not only the optimal position, but the optimal sparse kernel configurations. Also, the potential of the differentiable census transform for supervised depth estimation is yet to be investigated.

Bibliography

[1] *4D Light Field Benchmark — lightfield-analysis.uni-konstanz.de*. https://lightfield-analysis.uni-konstanz.de/. [Accessed 07-May-2023].

[2] Andrew Adams. *The (New) Stanford Light Field Archive*. http://lightfield.stanford.edu/lfs.html. [Accessed 05-Apr-2023]. 2008.

[3] Edward H Adelson and James R Bergen. "The plenoptic function and the elements of early vision". In: *Computational models of visual processing* 1 (1991), p. 8.

[4] Edward H. Adelson and John Y. A. Wang. "Single Lens Stereo with a Plenoptic Camera". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2 (1992), pp. 99–106. DOI: 10.1109/34.121783.

[5] Yuriy Anisimov, Jason Raphael Rambach, and Didier Stricker. "Nonlinear Optimization of Light Field Point Cloud". In: *Sensors* 22.3 (2022), p. 814. DOI: 10.3390/s22030814.

[6] Yuriy Anisimov and Didier Stricker. "Fast and Efficient Depth Map Estimation from Light Fields". In: *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*. IEEE Computer Society, 2017, pp. 337–346. DOI: 10.1109/3DV.2017.00046.

[7] Yuriy Anisimov, Oliver Wasenmüller, and Didier Stricker. "A Compact Light Field Camera for Real-Time Depth Estimation". In: *Computer Analysis of Images and Patterns - 18th International Conference, CAIP 2019, Salerno, Italy, September 3-5, 2019, Proceedings, Part I*. Ed. by Mario Vento and Gennaro Percannella. Vol. 11678. Lecture Notes in Computer Science. Springer, 2019, pp. 52–63. DOI: 10.1007/978-3-030-29888-3_5.

[8] Yuriy Anisimov, Oliver Wasenmüller, and Didier Stricker. "Rapid Light Field Depth Estimation with Semi-Global Matching". In: *15th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP 2019, Cluj-Napoca, Romania, September 5-7, 2019*. Ed. by Sergiu Nedevschi, Rodica Potolea, and Radu Razvan Slavescu. IEEE, 2019, pp. 109–116. DOI: 10.1109/ICCP48234.2019.8959680.

[9] K. S. Arun, Thomas S. Huang, and Steven D. Blostein. "Least-Squares Fitting of Two 3-D Point Sets". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 9.5 (1987), pp. 698–700. DOI: 10.1109/TPAMI.1987.4767965.

[10] Michael Barr and Anthony J Massa. *Programming Embedded Systems*. en. 2nd ed. Sebastopol, CA: O'Reilly Media, 2006.

[11] Bryce E Bayer. *Color imaging array*. US Patent 3,971,065. 1976.

[12] Commandant Benoit. "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrésa un systeme d'équations linéaires en nombre inférieura celui des inconnues. Application de la méthodea la résolution d'un systeme défini d'équations linéaires (Procédé du Commandant Cholesky)". In: *Bulletin géodésique* 2.1 (1924), pp. 67–77.

[13] Filippo Bergamasco et al. "Adopting an unconstrained ray model in light-field cameras for 3D shape reconstruction". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 3003–3012. DOI: 10.1109/CVPR.2015.7298919.

[14] Patrick Bertram et al. "Development of a context-aware assistive system for manual repair processes-a combination of probabilistic and deterministic approaches". In: *Procedia Manufacturing* 51 (2020), pp. 598–604.

[15] Julian Besag. "Statistical analysis of non-lattice data". In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 24.3 (1975), pp. 179–195. DOI: 10.2307/2987782.

[16] Mario Bettini. *Apiaria universae philosopiae mathematicae*. 1642.

[17] Yunsu Bok, Hae-Gon Jeon, and In So Kweon. "Geometric Calibration of Micro-Lens-Based Light Field Cameras Using Line Features". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.2 (2017), pp. 287–300. DOI: 10.1109/TPAMI.2016.2541145.

[18] Robert C. Bolles, H. Harlyn Baker, and David H. Marimont. "Epipolar-plane image analysis: An approach to determining structure from motion". In: *Int. J. Comput. Vis.* 1.1 (1987), pp. 7–55. DOI: 10.1007/BF00128525.

[19] Mohammad Reza Bonyadi and Zbigniew Michalewicz. "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review". In: *Evol. Comput.* 25.1 (2017), pp. 1–54. DOI: 10.1162/EVCO_r_00180.

[20] Jean-Yves Bouguet. *Camera Calibration Toolbox for Matlab*. en. CaltechDATA, 2022. DOI: 10.22002/D1.20164.

[21] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast Approximate Energy Minimization via Graph Cuts". In: *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*. IEEE Computer Society, 1999, pp. 377–384. DOI: 10.1109/ICCV.1999.791245.

[22] Michael H. Brill. "Trichromatic Theory". In: *Computer Vision, A Reference Guide*. 2014, pp. 827–829. DOI: 10.1007/978-0-387-31439-6_453.

[23] Duane C Brown. "Decentering distortion of lenses". In: *Photogrammetric Engineering and Remote Sensing* (32 1966).

[24] G. Buchsbaum. "A spatial processor model for object colour perception". In: *Journal of the Franklin Institute* 310.1 (1980), pp. 1–26. ISSN: 0016-0032. DOI: https://doi.org/10.1016/0016-0032(80)90058-7.

[25] John F. Canny. "A Computational Approach to Edge Detection". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[26] Chaur-Chin Chen and Hsueh-Ting Chu. "Similarity Measurement Between Images". In: *29th Annual International Computer Software and Applications Conference, COMPSAC 2005, Edinburgh, Scotland, UK, July 25-28, 2005. Volume 2*. IEEE Computer Society, 2005, pp. 41–42. DOI: 10.1109/COMPSAC.2005.140.

[27] George Chen et al. "Light field duality: concept and applications". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 2002, Hong Kong, China, November 11-13, 2002*. Ed. by Jiaoying Shi et al. ACM, 2002, pp. 9–16. DOI: 10.1145/585740.585743.

[28] Jiaxin Chen, Shuo Zhang, and Youfang Lin. "Attention-based Multi-Level Fusion Network for Light Field Depth Estimation". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 1009–1017.

[29] Yang Chen, Martin Alain, and Aljosa Smolic. "Fast and Accurate Optical Flow based Depth Map Estimation from Light Fields". In: *CoRR* abs/2008.04673 (2020). arXiv: 2008.04673.

[30] Gabriella Csurka et al. "Characterizing the uncertainty of the fundamental matrix". In: *Computer vision and image understanding* 68.1 (1997), pp. 18–36. DOI: 10.1006/cviu.1997.0531.

[31] Lukasz Dabala et al. "Efficient Multi-image Correspondences for Online Light Field Video Processing". In: *Comput. Graph. Forum* 35.7 (2016), pp. 401–410. DOI: 10.1111/cgf.13037.

[32] Sven Dupré. "Inside the camera obscura: Kepler's experiment and theory of optical imagery". In: *Early Science and Medicine* 13.3 (2008), pp. 219–244. DOI: 10.1163/157338208x285026.

[33] Qingbin Fan et al. "Trilobite-inspired neural nanophotonic light-field camera with extreme depth-of-field". In: *Nature Communications* 13.1 (2022). DOI: 10.1038/s41467-022-29568-y.

[34] Olivier D. Faugeras, Quang-Tuan Luong, and Stephen J. Maybank. "Camera Self-Calibration: Theory and Experiments". In: *Computer Vision - ECCV'92, Second European Conference on Computer Vision, Santa Margherita Ligure, Italy, May 19-22, 1992, Proceedings*. Ed. by Giulio Sandini. Vol. 588. Lecture Notes in Computer Science. Springer, 1992, pp. 321–334. DOI: 10.1007/3-540-55426-2_37.

[35] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (1981), pp. 381–395. DOI: 10.1145/358669.358692.

[36] Wolfgang Förstner and Eberhard Gülch. "A fast operator for detection and precise location of distinct points, corners and centres of circular features". In: *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*. Vol. 6. Interlaken. 1987, pp. 281–305.

[37] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[38] Andrei Gershun. "The light field". In: *Journal of Mathematics and Physics* 18.1-4 (1939), pp. 51–151.

[39] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Third Edition*. Johns Hopkins University Press, 1996. ISBN: 978-0-8018-5414-9.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[41] Walter Greiner. *Quantum Mechanics*. Springer Berlin Heidelberg, 2001. DOI: 10.1007/978-3-642-56826-8.

[42] István Haller and Sergiu Nedevschi. "Design of Interpolation Functions for Subpixel-Accuracy Stereo-Vision Systems". In: *IEEE Trans. Image Process.* 21.2 (2012), pp. 889–898. DOI: 10.1109/TIP.2011.2163163.

[43] Song Han, Huizi Mao, and William J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. 2015. DOI: 10.48550/ARXIV.1510.00149.

[44] Robert M. Haralick, Stanley R. Sternberg, and Xinhua Zhuang. "Image Analysis Using Mathematical Morphology". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 9.4 (1987), pp. 532–550. DOI: 10.1109/TPAMI.1987.4767941.

[45] Christopher G. Harris and Mike Stephens. "A Combined Corner and Edge Detector". In: *Proceedings of the Alvey Vision Conference, AVC 1988, Manchester, UK, September, 1988*. Ed. by Christopher J. Taylor. Alvey Vision Club, 1988, pp. 1–6. DOI: 10.5244/C.2.23.

[46] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. ISBN: 9780511811685. DOI: 10.1017/cbo9780511811685.

[47] Li WangDong-Chen He. "Texture classification using texture spectrum". In: *Pattern Recognit.* 23.8 (1990), pp. 905–910. DOI: 10.1016/0031-3203(90)90135-8.

[48] Steve Heath. *Embedded Systems Design*. en. 2nd ed. London, England: Newnes, 2002. ISBN: 9780750632379.

[49] Stefan Heber and Thomas Pock. "Convolutional Networks for Shape from Light Field". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 3746–3754. DOI: 10.1109/CVPR.2016.407.

[50] Stefan Heber, Wei Yu, and Thomas Pock. "Neural EPI-Volume Networks for Shape from Light Field". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2271–2279. DOI: 10.1109/ICCV.2017.247.

[51] Robert Hirsch. *Seizing the light: a social & aesthetic history of photography*. Routledge, 2017. ISBN: 978-11-3894-425-1.

[52] Heiko Hirschmüller. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.2 (2008), pp. 328–341. DOI: 10.1109/TPAMI.2007.1166.

[53] Heiko Hirschmüller, Peter R. Innocent, and Jonathan M. Garibaldi. "Real-Time Correlation-Based Stereo Vision with Reduced Border Errors". In: *Int. J. Comput. Vis.* 47.1-3 (2002), pp. 229–246. DOI: 10.1023/A:1014554110407.

[54] Heiko Hirschmüller and Daniel Scharstein. "Evaluation of Cost Functions for Stereo Matching". In: *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. IEEE Computer Society, 2007. DOI: 10.1109/CVPR.2007.383248.

[55] Katrin Honauer et al. "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*. Ed. by Shang-Hong Lai et al. Vol. 10113. Lecture Notes in Computer Science. Springer, 2016, pp. 19–34. DOI: 10.1007/978-3-319-54187-7_2.

[56] Berthold K. P. Horn and Brian G. Schunck. "Determining Optical Flow". In: *Artif. Intell.* 17.1-3 (1981), pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2.

[57] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[58] Qiuxia Hou and Cheolkon Jung. "Occlusion Robust Light Field Depth Estimation Using Segmentation Guided Bilateral Filtering". In: *19th IEEE International Symposium on Multimedia, ISM 2017, Taichung, Taiwan, December 11-13, 2017*. IEEE Computer Society, 2017, pp. 14–18. DOI: 10.1109/ISM.2017.13.

[59] Xiaoyan Hu and Philippos Mordohai. "A Quantitative Evaluation of Confidence Measures for Stereo Vision". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (2012), pp. 2121–2133. DOI: 10.1109/TPAMI.2012.46.

[60] Baoru Huang et al. "H-Net: Unsupervised Attention-based Stereo Depth Estimation Leveraging Epipolar Geometry". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022*. IEEE, 2022, pp. 4459–4466. DOI: 10.1109/CVPRW56347.2022.00492.

[61] Chao-Tsung Huang. "Robust Pseudo Random Fields for Light-Field Stereo Matching". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 11–19. DOI: 10.1109/ICCV.2017.11.

[62] Zhicong Huang et al. "Fast Light-field Disparity Estimation with Multi-disparity-scale Cost Aggregation". In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 6300–6309. DOI: 10.1109/ICCV48922.2021.00626.

[63] Katsushi Ikeuchi, ed. *Computer Vision, A Reference Guide*. Springer, 2014. ISBN: 978-0-387-30771-8. DOI: 10.1007/978-0-387-31439-6.

[64] Stephen S. Intille and Aaron F. Bobick. "Disparity-Space Images and Large Occlusion Stereo". In: *Computer Vision - ECCV'94, Third European Conference on Computer Vision, Stockholm, Sweden, May 2-6, 1994, Proceedings, Volume II*. Ed. by Jan-Olof Eklundh. Vol. 801. Lecture Notes in Computer Science. Springer, 1994, pp. 179–186. DOI: 10.1007/BFb0028349.

[65] Taisei Iwatsuki, Keita Takahashi, and Toshiaki Fujii. "Unsupervised disparity estimation from light field using plug-and-play weighted warping loss". In: *Signal Process. Image Commun.* 107 (2022), p. 116764. DOI: 10.1016/j.image.2022.116764.

[66] Hae-Gon Jeon et al. "Accurate depth map estimation from a lenslet light field camera". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 1547–1555. DOI: 10.1109/CVPR.2015.7298762.

[67] Hae-Gon Jeon et al. "Depth from a Light Field Image with Learning-Based Matching Costs". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 41.2 (2019), pp. 297–310. DOI: 10.1109/TPAMI.2018.2794979.

[68] Yue Ji and Jun Wu. "Calibration method of light-field camera for photogrammetry application". In: *Measurement* 148 (2019), p. 106943. DOI: 10.1016/j.measurement.2019.106943.

[69] Jing Jin and Junhui Hou. "Occlusion-Aware Unsupervised Learning of Depth From 4-D Light Fields". In: *IEEE Trans. Image Process.* 31 (2022), pp. 2216–2228. DOI: 10.1109/TIP.2022.3154288.

[70] Ole Johannsen, Antonin Sulc, and Bastian Goldluecke. "What Sparse Light Field Coding Reveals about Scene Structure". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 3262–3270. DOI: 10.1109/CVPR.2016.355.

[71] Neel Joshi and Larry Zitnick. "Micro-baseline stereo". In: *Technical Report MSR-TR-2014–73* (2014), p. 8.

[72] Daniel Hernández Juárez et al. "Embedded Real-time Stereo Estimation via Semi-Global Matching on the GPU". In: *International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA*. Ed. by Michelle Connolly. Vol. 80. Procedia Computer Science. Elsevier, 2016, pp. 143–153. DOI: 10.1016/j.procs.2016.05.305.

[73] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. "Local Binary Convolutional Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 4284–4293. DOI: 10.1109/CVPR.2017.456.

[74] Yun-Suk Kang, Cheon Lee, and Yo-Sung Ho. "An Efficient Rectification Algorithm for Multi-View Images in Parallel Camera Array". In: *2008 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*. 2008, pp. 61–64. DOI: 10.1109/3DTV.2008.4547808.

[75] Changil Kim et al. "Scene reconstruction from high spatio-angular resolution light fields". In: *ACM Trans. Graph.* 32.4 (2013), 73:1–73:12. DOI: 10.1145/2461912.2461926.

[76] Ron Kimmel. "Demosaicing: image reconstruction from color CCD samples". In: *IEEE Trans. Image Process.* 8.9 (1999), pp. 1221–1228. DOI: 10.1109/83.784434.

[77] Reinhard Koch. "Depth Estimation". In: *Computer Vision: A Reference Guide*. Ed. by Katsushi Ikeuchi. Boston, MA: Springer US, 2014, pp. 183–186. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_125.

[78] 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*. Ed. by Anders Heyden et al. Vol. 2352. Lecture Notes in Computer Science. Springer, 2002, pp. 82–96. DOI: 10.1007/3-540-47977-5_6.

[79] Johannes Kopf et al. "Joint bilateral upsampling". In: *ACM Trans. Graph.* 26.3 (2007), p. 96. DOI: 10.1145/1276377.1276497.

[80] Sanjeev J. Koppal. "Lambertian Reflectance". In: *Computer Vision, A Reference Guide*. 2014, pp. 441–443. DOI: 10.1007/978-0-387-31439-6_534.

[81] Bernd Krolla, Maximilian Diebold, and Didier Stricker. "Light Field from Smartphone-Based Dual Video". In: *Computer Vision - ECCV 2014 Workshops - Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II*. Ed. by Lourdes Agapito, Michael M. Bronstein, and Carsten Rother. Vol. 8926. Lecture Notes in Computer Science. Springer, 2014, pp. 600–610. DOI: 10.1007/978-3-319-16181-5_46.

[82] Jae Young Lee and Rae-Hong Park. "Complex-Valued Disparity: Unified Depth Model of Depth from Stereo, Depth from Focus, and Depth from Defocus Based on the Light Field Gradient". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 43.3 (2021), pp. 830–841. DOI: 10.1109/TPAMI.2019.2946159.

[83] Jae Young Lee and Rae-Hong Park. "Depth Estimation From Light Field by Accumulating Binary Maps Based on Foreground-Background Separation". In: *IEEE J. Sel. Top. Signal Process.* 11.7 (2017), pp. 955–964. DOI: 10.1109/JSTSP.2017.2747154.

[84] Titus Leistner et al. "Learning to Think Outside the Box: Wide-Baseline Light Field Depth Estimation with EPI-Shift". In: *2019 International Conference on 3D Vision, 3DV 2019, Québec City, QC, Canada, September 16-19, 2019*. IEEE, 2019, pp. 249–257. DOI: 10.1109/3DV.2019.00036.

[85] Titus Leistner et al. "Towards Multimodal Depth Estimation from Light Fields". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 12943–12951. DOI: 10.1109/CVPR52688.2022.01261.

[86] Marc Levoy and Pat Hanrahan. "Light Field Rendering". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*. Ed. by John Fujii. ACM, 1996, pp. 31–42. DOI: 10.1145/237170.237199.

[87] Kunyuan Li et al. "Self-Supervised Light Field Depth Estimation Using Epipolar Plane Images". In: *International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021*. IEEE, 2021, pp. 731–740. DOI: 10.1109/3DV53792.2021.00082.

[88] Peng Li et al. "OPAL: Occlusion Pattern Aware Loss for Unsupervised Light Field Disparity Estimation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 46.2 (2024), pp. 681–694. DOI: 10.1109/TPAMI.2023.3296600.

[89] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Advances in Pattern Recognition. Springer, 2009. ISBN: 978-1-84800-278-4. DOI: 10.1007/978-1-84800-279-1.

[90] "Hamming Distance". In: *Encyclopedia of Biometrics*. Ed. by Stan Z. Li and Anil Jain. Boston, MA: Springer US, 2009, pp. 668–668. ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_956.

[91] Tailin Liang et al. "Pruning and quantization for deep neural network acceleration: A survey". In: *Neurocomputing* 461 (2021), pp. 370–403. DOI: 10.1016/j.neucom.2021.07.045.

[92] Haiting Lin et al. "Depth Recovery from Light Field Using Focal Stack Symmetry". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 3451–3459. DOI: 10.1109/ICCV.2015.394.

[93] Fei Liu et al. "High quality depth map estimation of object surface from light-field images". In: *Neurocomputing* 252 (2017), pp. 3–16. DOI: 10.1016/j.neucom.2016.09.136.

[94] Maziar Loghman and Joohee Kim. "SGM-based dense disparity estimation using adaptive Census transform". In: *International Conference on Connected Vehicles and Expo, ICCVE 2012, Las Vegas, NV, USA, December 2-6, 2013*. IEEE, 2013, pp. 592–597. DOI: 10.1109/ICCVE.2013.6799860.

[95] Manolis I.A. Lourakis and Rachid Deriche. *Camera Self-Calibration Using the Singular Value Decomposition of the Fundamental Matrix: From Point Correspondences to 3D Measurements*. Tech. rep. RR-3748. INRIA, 1999.

[96] David G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*. IEEE Computer Society, 1999, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.

[97] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24-28, 1981*. Ed. by Patrick J. Hayes. William Kaufmann, 1981, pp. 674–679.

[98] Yaoxiang Luo et al. "EPI-Patch Based Convolutional Neural Network for Depth Estimation on 4D Light Field". In: *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part III*. Ed. by Derong Liu et al. Vol. 10636. Lecture Notes in Computer Science. Springer, 2017, pp. 642–652. DOI: 10.1007/978-3-319-70090-8_65.

[99] Raman Maini and Himanshu Aggarwal. "A Comprehensive Review of Image Enhancement Techniques". In: *CoRR* abs/1003.4053 (2010). arXiv: 1003.4053.

[100] Simon Meister, Junhwa Hur, and Stefan Roth. "Unflow: Unsupervised learning of optical flow with a bidirectional census loss". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018. DOI: 10.1609/aaai.v32i1.12276.

[101] Moritz Menze and Andreas Geiger. "Object scene flow for autonomous vehicles". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 3061–3070. DOI: 10.1109/CVPR.2015.7298925.

[102] Yoshiki Mizukami et al. "Sub-pixel disparity search for binocular stereo vision". In: *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*. IEEE Computer Society, 2012, pp. 364–367.

[103] Rebekah Modrak and Bill Anthes. *Reframing Photography*. Routledge, 2010. DOI: 10.4324/9780203847596.

[104] Jorge J. Moré. "The Levenberg-Marquardt algorithm: Implementation and theory". In: *Numerical Analysis*. Ed. by G. A. Watson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116. ISBN: 978-3-540-35972-2.

[105] Jamuna S Murthy et al. "ObjectDetect: A Real-Time Object Detection Framework for Advanced Driver Assistant Systems Using YOLOv5". In: *Wireless Communications and Mobile Computing* 2022 (2022). DOI: 10.1155/2022/9444360.

[106] Julia Navarro and Antoni Buades. "Robust and Dense Depth Estimation for Light Field Images". In: *IEEE Trans. Image Process.* 26.4 (2017), pp. 1873–1886. DOI: 10.1109/TIP.2017.2666041.

[107] Joseph Needham. *Science and civilisation in China physics and physical technology: Volume 4: Physics part 1*. Cambridge, England: Cambridge University Press, 1962. ISBN: 9780521058025.

[108] Alessandro Neri, Marco Carli, and Federica Battisti. "A multi-resolution approach to depth field estimation in dense image arrays". In: *2015 IEEE International Conference on Image Processing, ICIP 2015, Quebec City, QC, Canada, September 27-30, 2015*. IEEE, 2015, pp. 3358–3362. DOI: 10.1109/ICIP.2015.7351426.

[109] MEJ Newman. "Power laws, Pareto distributions and Zipf's law". In: *Contemporary Physics* 46.5 (2005), pp. 323–351. DOI: 10.1080/00107510500052444.

[110] Phillip A. Newman and Vincent E. Rible. "Pinhole Array Camera for Integrated Circuits". In: *Appl. Opt.* 5.7 (1966), pp. 1225–1228. DOI: 10.1364/AO.5.001225.

[111] Ren Ng et al. *Light Field Photography with a Hand-held Plenoptic Camera*. Research Report CSTR 2005-02. Stanford university, 2005, Stanford University Computer Science Tech Report.

[112] Charles-Antoine Noury, Céline Teulière, and Michel Dhome. "Light-Field Camera Calibration from Raw Images". In: *2017 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2017, Sydney, Australia, November 29 - December 1, 2017*. IEEE, 2017, pp. 1–8. DOI: 10.1109/DICTA.2017.8227459.

[113] Sean G. P. O'Brien et al. "Calibrating Light-Field Cameras Using Plenoptic Disc Features". In: *2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018*. IEEE Computer Society, 2018, pp. 286–294. DOI: 10.1109/3DV.2018.00041.

[114] Masatoshi Okutomi and Takeo Kanade. "A Multiple-Baseline Stereo". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 15.4 (1993), pp. 353–363. DOI: 10.1109/34.206955.

[115] James O'Neill. "An Overview of Neural Network Compression". In: *CoRR* abs/2006.03669 (2020). arXiv: 2006.03669.

[116] *OpenVX - Portable, Power-efficient Vision Processing — khronos.org*. https://www.khronos.org/openvx/. [Accessed 05-May-2023].

[117] A. Orth et al. "Optical fiber bundles: Ultra-slim light field imaging probes". In: *Science Advances* 5.4 (2019). DOI: 10.1126/sciadv.aav1555.

[118] Théodore Papadopoulo and Manolis IA Lourakis. "Estimating the jacobian of the singular value decomposition: Theory and applications". In: *Computer Vision-ECCV 2000: 6th European Conference on Computer Vision Dublin, Ireland, June 26–July 1, 2000 Proceedings, Part I 6*. Springer. 2000, pp. 554–570. DOI: 10.1007/3-540-45054-8_36.

[119] Emanuel Parzen. "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[120] Jiayong Peng et al. "Unsupervised Depth Estimation from Light Field Using a Convolutional Neural Network". In: *2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018*. IEEE Computer Society, 2018, pp. 295–303. DOI: 10.1109/3DV.2018.00042.

[121] Jiayong Peng et al. "Zero-Shot Depth Estimation From Light Field Using A Convolutional Neural Network". In: *IEEE Trans. Computational Imaging* 6 (2020), pp. 682–696. DOI: 10.1109/TCI.2020.2967148.

[122] Cristiano Premebida, Gledson Melotti, and Alireza Asvadi. "RGB-D object classification for autonomous driving perception". In: *RGB-D Image Analysis and Processing* (2019), pp. 377–395. DOI: 10.1007/978-3-030-28603-3_17.

[123] Yanwen Qin, Xin Jin, and Qionghai Dai. "GPU-based depth estimation for light field images". In: *2017 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2017, Xiamen, China, November 6-9, 2017*. IEEE, 2017, pp. 640–645. DOI: 10.1109/ISPACS.2017.8266556.

[124] Yanwen Qin et al. "Enhanced depth estimation for hand-held light field cameras". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, 2017, pp. 2032–2036. DOI: 10.1109/ICASSP.2017.7952513.

[125] Mohammad Rastegari et al. "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks". In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*. Ed. by Bastian Leibe et al. Vol. 9908. Lecture Notes in Computer Science. Springer, 2016, pp. 525–542. DOI: 10.1007/978-3-319-46493-0_32.

[126] Russell Reed. "Pruning algorithms-a survey". In: *IEEE Trans. Neural Networks* 4.5 (1993), pp. 740–747. DOI: 10.1109/72.248452.

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

[128] Jérôme Revaud et al. "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow". In: *CoRR* abs/1501.02565 (2015). arXiv: 1501.02565.

[129] O Rodrigues. "On the geometrical laws that govern the displacements of a solid system in space, and on the change of coordinates resulting from these displacements considered independently of the causes that can produce them". In: *J Math Pures Appl* 5 (1840), pp. 380–440.

[130] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: *Int. J. Comput. Vis.* 40.2 (2000), pp. 99–121. DOI: 10.1023/A:1026543900054.

[131] Neus Sabater et al. "Dataset and Pipeline for Multi-view Light-Field Video". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 1743–1753. DOI: 10.1109/CVPRW.2017.221.

[132] MRV Sahyun. "Mechanisms in photographic chemistry". In: *Journal of Chemical Education* 51.2 (1974), p. 72. DOI: 10.1021/ed051p72.

[133] Daniel Scharstein and Chris Pal. "Learning conditional random fields for stereo". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[134] Daniel Scharstein and Richard Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms". In: *Int. J. Comput. Vis.* 47.1-3 (2002), pp. 7–42. DOI: 10.1023/A:1014573219977.

[135] Steven A Shafer. "Using color to separate reflection components". In: *Color Research & Application* 10.4 (1985), pp. 210–218. DOI: 10.1002/col.5080100409.

[136] Hao Sheng et al. "Occlusion-aware depth estimation for light field using multi-orientation EPIs". In: *Pattern Recognit.* 74 (2018), pp. 587–599. DOI: 10.1016/j.patcog.2017.09.010.

[137] Jianbo Shi and Carlo Tomasi. "Good features to track". In: *Conference on Computer Vision and Pattern Recognition, CVPR 1994, 21-23 June, 1994, Seattle, WA, USA*. IEEE, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[138] Changha Shin et al. "EPINET: A Fully-Convolutional Neural Network Using Epipolar Geometry for Depth From Light Field Images". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4748–4757. DOI: 10.1109/CVPR.2018.00499.

[139] Lipeng Si and Qing Wang. "Dense Depth-Map Estimation and Geometry Inference from Light Fields via Global Optimization". In: *Computer Vision - ACCV 2016 - 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III*. Ed. by Shang-Hong Lai et al. Vol. 10113. Lecture Notes in Computer Science. Springer, 2016, pp. 83–98. DOI: 10.1007/978-3-319-54187-7_6.

[140] A Mark Smith. *Alhacen's Theory of Visual Perception: A Critical Edition, with English Translation and Commentary, of the First Three Books of Alhacen's De Aspectibus, the Medieval Latin Version of Ibn Al-Haytham's Kitab Al-Manazir*. Vol. 1. American Philosophical Society, 2001. ISBN: 9780871699145.

[141] Steven W Smith et al. *The scientist and engineer's guide to digital signal processing*. 1997.

[142] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (2014).

[143] Michael Strecke, Anna Alperovich, and Bastian Goldluecke. "Accurate Depth and Normal Maps from Occlusion-Aware Focal Stack Symmetry". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 2529–2537. DOI: 10.1109/CVPR.2017.271.

[144] Jian Sun, Nanning Zheng, and Heung-Yeung Shum. "Stereo Matching Using Belief Propagation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.7 (2003), pp. 787–800. DOI: 10.1109/TPAMI.2003.1206509.

[145] Xing Sun et al. "Data-driven light field depth estimation using deep Convolutional Neural Networks". In: *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*. IEEE, 2016, pp. 367–374. DOI: 10.1109/IJCNN.2016.7727222.

[146] Xufu Sun et al. "Blind Calibration for Focused Plenoptic Cameras". In: *IEEE International Conference on Multimedia and Expo, ICME 2019, Shanghai, China, July 8-12, 2019*. IEEE, 2019, pp. 115–120. DOI: 10.1109/ICME.2019.00028.

[147] I.E. Sutherland. "Three-dimensional data input by tablet". In: *Proceedings of the IEEE* 62.4 (1974), pp. 453–461. DOI: 10.1109/PROC.1974.9449.

[148] Satoshi Suzuki and Keiichi Abe. "Topological structural analysis of digitized binary images by border following". In: *Comput. Vis. Graph. Image Process.* 30.1 (1985), pp. 32–46. DOI: 10.1016/0734-189X(85)90016-7.

[149] Michael W. Tao et al. "Depth Estimation for Glossy Surfaces with Light-Field Cameras". In: *Computer Vision - ECCV 2014 Workshops - Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II*. Ed. by Lourdes Agapito, Michael M. Bronstein, and Carsten Rother. Vol. 8926. Lecture Notes in Computer Science. Springer, 2014, pp. 533–547. DOI: 10.1007/978-3-319-16181-5_41.

[150] Michael W. Tao et al. "Depth from Combining Defocus and Correspondence Using Light-Field Cameras". In: *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*. IEEE Computer Society, 2013, pp. 673–680. DOI: 10.1109/ICCV.2013.89.

[151] Michael W. Tao et al. "Shape Estimation from Shading, Defocus, and Correspondence Using Light-Field Angular Coherence". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.3 (2017), pp. 546–560. DOI: 10.1109/TPAMI.2016.2554121.

[152] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features*. Tech. rep. International Journal of Computer Vision, 1991.

[153] Takayuki Tomioka et al. "Depth Map Estimation Using Census Transform for Light Field Cameras". In: *IEICE Trans. Inf. Syst.* 100-D.11 (2017), pp. 2711–2720. DOI: 10.1587/transinf.2017EDP7052.

[154] Bill Triggs et al. "Bundle Adjustment - A Modern Synthesis". In: *Vision Algorithms: Theory and Practice, International Workshop on Vision Algorithms, held during ICCV '99, Corfu, Greece, September 21-22, 1999, Proceedings*. Ed. by Bill Triggs, Andrew Zisserman, and Richard Szeliski. Vol. 1883. Lecture Notes in Computer Science. Springer, 1999, pp. 298–372. DOI: 10.1007/3-540-44480-7_21.

[155] Harit P Trivedi. "Can multiple views make up for lack of camera registration?" In: *Image and Vision Computing* 6.1 (1988), pp. 29–32.

[156] Yu-Ju Tsai et al. "Attention-Based View Selection Networks for Light-Field Disparity Estimation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.07 (2020), pp. 12095–12103. DOI: 10.1609/aaai.v34i07.6888.

[157] Vaibhav Vaish et al. "Using Plane + Parallax for Calibrating Dense Camera Arrays". In: *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), with CD-ROM, 27 June - 2 July 2004, Washington, DC, USA*. IEEE Computer Society, 2004, pp. 2–9. DOI: 10.1109/CVPR.2004.257.

[158] Kartik Venkataraman et al. "*PiCam*: an ultra-thin high performance monolithic camera array". In: *ACM Trans. Graph.* 32.6 (2013), 166:1–166:13. DOI: 10.1145/2508363.2508390.

[159] Nick Waltham. "CCD and CMOS sensors". In: *Observing Photons in Space: A Guide to Experimental Space Astronomy*. Ed. by Martin C. E. Huber et al. New York, NY: Springer New York, 2013, pp. 423–442. ISBN: 978-1-4614-7804-1. DOI: 10.1007/978-1-4614-7804-1_23.

[160] Ting-Chun Wang, Alexei A. Efros, and Ravi Ramamoorthi. "Occlusion-Aware Depth Estimation Using Light-Field Cameras". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 3487–3495. DOI: 10.1109/ICCV.2015.398.

[161] Yang Wang et al. "UnOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 8071–8081. DOI: 10.1109/CVPR.2019.00826.

[162] Yangfan Wang et al. "Recent advances in 3D object detection based on RGB-D: A survey". In: *Displays* 70 (2021), p. 102077. DOI: 10.1016/j.displa.2021.102077.

[163] Yingqian Wang et al. "Disentangling Light Fields for Super-Resolution and Disparity Estimation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.1 (2023), pp. 425–443. DOI: 10.1109/TPAMI.2022.3152488.

[164] Sven Wanner and Bastian Goldluecke. "Globally consistent depth labeling of 4D light fields". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*. IEEE Computer Society, 2012, pp. 41–48. DOI: 10.1109/CVPR.2012.6247656.

[165] Sven Wanner and Bastian Goldluecke. "Variational Light Field Analysis for Disparity Estimation and Super-Resolution". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 36.3 (2014), pp. 606–619. DOI: 10.1109/TPAMI.2013.147.

[166] Henry S Warren. *Hacker's Delight*. en. 2nd ed. Boston, MA: Addison-Wesley Educational, 2012. ISBN: 978-0321842688.

[167] Oliver Wasenmüller, Gabriele Bleser, and Didier Stricker. "Combined Bilateral Filter for Enhanced Real-time Upsampling of Depth Images". In: *VISAPP 2015 - Proceedings of the 10th International Conference on Computer Vision Theory and Applications, Volume 1, Berlin, Germany, 11-14 March, 2015*. Ed. by José Braz, Sebastiano Battiato, and Francisco H. Imai. SciTePress, 2015, pp. 5–12. DOI: 10.5220/0005234800050012.

[168] John Werge. *The evolution of photography: with a chronological record of discoveries, inventions, etc., contributions to photographic literature, and personal reminiscences extending over forty years*. Piper & Carter and J. Werge, 1890.

[169] Charles Wheatstone. "XVIII. Contributions to the physiology of vision. —Part the first. On some remarkable, and hitherto unobserved, phenomena of binocular vision". In: *Philosophical Transactions of the Royal Society of London* 128 (1838), pp. 371–394. DOI: 10.1098/rstl.1838.0019.

[170] Williem and In Kyu Park. "Robust Light Field Depth Estimation for Noisy Scene with Occlusion". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 4396–4404. DOI: 10.1109/CVPR.2016.476.

[171] Williem, In Kyu Park, and Kyoung Mu Lee. "Robust Light Field Depth Estimation Using Occlusion-Noise Aware Data Costs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.10 (2018), pp. 2484–2497. DOI: 10.1109/TPAMI.2017.2746858.

[172] Yichao Xu et al. "Camera array calibration for light field acquisition". In: *Frontiers Comput. Sci.* 9.5 (2015), pp. 691–702. DOI: 10.1007/s11704-015-4237-4.

[173] Zhe Xu et al. "On-road multiple obstacles detection using color images and LiDAR point clouds". In: *Seventh International Conference on Optical and Photonic Engineering (icOPEN 2019)*. Vol. 11205. SPIE. 2019, pp. 276–282. DOI: 10.1117/12.2541656.

[174] Qingxiong Yang et al. "Spatial-Depth Super Resolution for Range Images". In: *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. IEEE Computer Society, 2007. DOI: 10.1109/CVPR.2007.383211.

[175] Matt Young. "Pinhole optics." In: *Applied optics* 10 12 (1971), pp. 2763–2767. DOI: 10.1364/AO.10.002763.

[176] Thomas Young. "II. The Bakerian Lecture. On the theory of light and colours". In: *Philosophical transactions of the Royal Society of London* 92 (1802), pp. 12–48. DOI: 10.1098/rstl.1802.0004.

[177] Kaan Yücer et al. "Depth from Gradients in Dense Light Fields for Object Reconstruction". In: *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*. IEEE Computer Society, 2016, pp. 249–257. DOI: 10.1109/3DV.2016.33.

[178] Ramin Zabih and John Woodfill. "Non-parametric Local Transforms for Computing Visual Correspondence". In: *Computer Vision - ECCV'94, Third European Conference on Computer Vision, Stockholm, Sweden, May 2-6, 1994, Proceedings, Volume II*. Ed. by Jan-Olof Eklundh. Vol. 801. Lecture Notes in Computer Science. Springer, 1994, pp. 151–158. DOI: 10.1007/BFb0028345.

[179] Bozena Zdaniuk. "Ordinary Least-Squares (OLS) Model". In: *Encyclopedia of Quality of Life and Well-Being Research*. Ed. by Alex C. Michalos. Dordrecht: Springer Netherlands, 2014, pp. 4515–4517. ISBN: 978-94-007-0753-5. DOI: 10.1007/978-94-007-0753-5_2008.

[180] N. Zeller et al. "Metric Calibration of a Focused Plenoptic Camera based on a 3D Calibration Target". In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III-3 (2016), pp. 449–456. DOI: 10.5194/isprs-annals-III-3-449-2016.

[181] Cha Zhang and Tsuhan Chen. "A Self-Reconfigurable Camera Array". In: *Proceedings of the 15th Eurographics Workshop on Rendering Techniques, Norköping, Sweden, June 21-23, 2004*. Ed. by Alexander Keller and Henrik Wann Jensen. Eurographics Association, 2004, pp. 243–254. DOI: 10.2312/EGWR/EGSR04/243-254.

[182] Shansi Zhang, Nan Meng, and Edmund Y. Lam. "Unsupervised Light Field Depth Estimation via Multi-view Feature Matching with Occlusion Prediction". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2023), pp. 1–1. DOI: 10.1109/TCSVT.2023.3305978.

[183] Shuo Zhang et al. "Robust depth estimation for light field via spinning parallelogram operator". In: *Comput. Vis. Image Underst.* 145 (2016), pp. 148–159. DOI: 10.1016/j.cviu.2015.12.007.

[184] Zhengyou Zhang. "A Flexible New Technique for Camera Calibration". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[185] Zhengyou Zhang. "Camera Calibration with One-Dimensional Objects". In: *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002, Proceedings, Part IV*. Ed. by Anders Heyden et al. Vol. 2353. Lecture Notes in Computer Science. Springer, 2002, pp. 161–174. DOI: 10.1007/3-540-47979-1_11.

[186] Changyin Zhou and Shree K. Nayar. "Computational Cameras: Convergence of Optics and Processing". In: *IEEE Trans. Image Process.* 20.12 (2011), pp. 3322–3340. DOI: 10.1109/TIP.2011.2171700.

[187] Ping Zhou et al. "Light field calibration and 3D shape measurement based on epipolar-space". In: *Opt. Express* 27.7 (2019), pp. 10171–10184. DOI: 10.1364/OE.27.010171.

[188] Wenhui Zhou et al. "Light-field flow: A subpixel-accuracy depth flow estimation with geometric occlusion model from a single light-field image". In: *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*. IEEE, 2017, pp. 1632–1636. DOI: 10.1109/ICIP.2017.8296558.

[189] Wenhui Zhou et al. "Unsupervised Monocular Depth Estimation From Light Field Image". In: *IEEE Trans. Image Process.* 29 (2020), pp. 1606–1617. DOI: 10.1109/TIP.2019.2944343.

[190] Hao Zhu, Qing Wang, and Jingyi Yu. "Occlusion-Model Guided Anti-occlusion Depth Estimation in Light Field". In: *IEEE J. Sel. Top. Signal Process.* 11.7 (2017), pp. 965–978. DOI: 10.1109/JSTSP.2017.2730818.

Curriculum Vitae

Yuriy Anisimov

Education

Jun 2015 **MEng (Electrical Engineering)**
Moscow State University Of Mechanical Engineering (MAMI)
Moscow, Russia

Sep 2014 - Feb 2015 **Intern**
University of Ulsan and Hyundai vehicle plant
Ulsan, South Korea

Jun 2013 **BEng (Electrical Engineering)**
Moscow State University Of Mechanical Engineering (MAMI)
Moscow, Russia

Technical Experience

Nov 2022 - now **Researcher**
Rheinland-Pfälzische Technische Universität
Kaiserslautern, Germany

Mar 2017 - Oct 2022 **Researcher**
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Kaiserslautern, Germany