3D Reconstruction Using Active Illumination

vorgelegt von

M. Sc. **Martin Grochulla**

geboren in Laurahütte, Polen

Dissertation

zur Erlangung des Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

des Fachbereichs 12, Mathematik und Informatik

der Philipps-Universität Marburg (Hochschulkennziffer 1180)

Erstgutachter Prof. Dr. Thorsten Thormählen

Zweitgutachter Prof. Dr. Hans-Peter Seidel

Einreichung am 08.07.2016

Prüfung am 05.12.2016

Marburg, Januar 2017

Abstract

In this thesis we present a pipeline for 3D model acquisition. Generating 3D models of real-world objects is an important task in computer vision with many applications, such as in 3D design, archaeology, entertainment, and virtual or augmented reality.

The contribution of this thesis is threefold: we propose a calibration procedure for the cameras, we describe an approach for capturing and processing photometric normals using gradient illuminations in the hardware set-up, and finally we present a multi-view photometric stereo 3D reconstruction method.

In order to obtain accurate results using multi-view and photometric stereo reconstruction, the cameras are calibrated geometrically and photometrically.

For acquiring data, a light stage is used. This is a hardware set-up that allows to control the illumination during acquisition. The procedure used to generate appropriate illuminations and to process the acquired data to obtain accurate photometric normals is described.

The core of the pipeline is a multi-view photometric stereo reconstruction method. In this method, we first generate a sparse reconstruction using the acquired images and computed normals. In the second step, the information from the normal maps is used to obtain a dense reconstruction of an object's surface. Finally, the reconstructed surface is filtered to remove artifacts introduced by the dense reconstruction step.

Kurzfassung

In dieser Arbeit präsentieren wir eine Pipeline zur Aufnahme von 3D-Modellen. Die Erzeugung von 3D-Modellen von realen Objekten ist eine wichtige Aufgabe in der Computer Vision mit zahlreichen Anwendungen, wie zum Beispiel 3D-Design, Archäologie und virtueller oder auch erweiterter Realität.
Die vorliegende Arbeit liefert drei Beiträge: wir stellen eine Methode zur Kalibrierung von Kameras vor, wir beschreiben einen Ansatz zur Aufnahme und Verarbeitung von photometrischen Normalen, die mit Hilfe von Gradientenbeleuchtung in der Hardwarekonfiguration aufgenommen worden sind, und schlieÃŸlich präsentieren wir eine Methode zur Stereorekonstruktion von 3D-Modellen aus photometrischen Daten und mehreren Perspektiven.
Um möglichst genaue Rekonstruktionsergebnisse aus mehreren Perspektiven mit Hilfe von photometrischen Daten zu erhalten, werden die Kameras sowohl geometrisch als auch photometrisch kalibriert.
Zur Datenaufnahme wird eine sogenannte “Light Stage” benutzt. Dies ist eine Hardwarekonfiguration, die es erlaubt die Beleuchtung während der Aufnahme vollsändig zu kontrollieren. Die verwendete Methode zur Erzeugung der Gradientenbeleuchtung und zur Verarbeitung der aufgenommenen Daten zur Erzeugung genauer photometrischer Normalen wird beschrieben.
Der Hauptbestandteil der Pipeline ist eine Methode zur Stereorekonstruktion aus photometrischen Daten und mehreren Perspektiven. Bei dieser Methode erzeugen wir zunächst eine grobe Rekonstruktion mit Hilfe der aufgenommenen Bilder und erzeugten Normalen. In einem zweiten Schritt werden die Normaleninformationen verwendet um eine vollständige Rekonstruktion der Objektoberfläche zu erhalten. Schließlich erfolgt eine Filterung der rekonstruierten Oberfläche, um Artefakte aus dem zweiten Rekonstruktionsschritt zu entfernen.

Acknowledgement

I would like to express my deepest gratitude to my advisor Prof. Dr. Thorsten Thormählen for being a tremendous mentor in the "Imaged-based 3D Scene Analysis" group at the MPI Informatik and the "Graphics and Multimedia Programming" group at the Philipps-University Marburg, for his supervision during that time and for his encouragement.

I would like to express my special appreciation to Prof. Dr. Hans-Peter Seidel and Prof. Dr. Thormählen for giving me the opportunity to pursue my PhD studies at the MPI Informatik, which provided an excellent and outstanding working environment.

I would also like to thank my co-authors Dr. Zhao Li, Prof. Dr. Thorsten Herfet from the Telecommunications Lab at Saarland university and Dr. Zhongjie Wang from MPI Informatik for their advice, support, and their contributions to the projects and publications we have been together working on.

I would like thank all other fellow researchers for their valuable discussions, comments and suggestion and all employees at the MPI Informatik, especially Sabine Budde and Ellen Fries for all the help in administration and Thomas Hirtz for his technical support.

I would like to thank Dr. Christian Kurz for valuable discussions concerning research, programming, software development as well as Dr. Bertram Somieski for countless engaging discussions about research, science, technology, and everything.

I would like to thank my family and parents for their help, support and care during all the years.

Finally, I would like to thank my dear wife Banafsheh for her support, encouragement and patience.

Contents

Chapter 1

Introduction

The acquisition of 3D models is one of the fundamental tasks in computer vision. 3D models are used in many different areas like design, archaeology, movies and computer games, virtual and augmented reality systems. Creating 3D models by artists is a slow, time consuming process. Hence, various methods for 3D reconstruction have been developed.

These methods are either active or passive reconstruction methods. Passive reconstruction methods only use the information that is provided by the object in the scene itself. Such approaches usually have simple hardware set-ups. But in this case an object can only be reconstructed partially. On the other hand, there are active reconstruction methods. Laser triangulation, structured light reconstruction, or photometric stereo reconstruction are examples of such methods. Here, the object is illuminated in a way such that a better reconstruction can be obtained. However, those methods use a more complex hardware setup, requiring a more elaborate calibration. Furthermore, structured light and especially laser triangulation techniques are rather slow.

Methods can also be classified into sparse and dense reconstruction methods. Passive methods typically rely on the available information of an object only. This leads to sparse reconstructions. Using surface fitting algorithms dense reconstructions can be obtained from sparse ones. However, those algorithms simply interpolate the data from the sparse reconstruction and create low detail reconstructions. On the other hand, dense reconstructions are typically obtained from active methods. Such methods obtain more information about parts of the objects that passive methods have problems with.

Furthermore, methods range from interactive methods to fully automatic ones. Interactive methods have the advantage that the user can guide the reconstruction process and improve the 3D model where it is not satisfactory. However, in order to enable interac-

tivity the methods have to be efficient and fast which rules out computationally expensive approaches and hence reduce the quality and accuracy of the result. Fully automatic methods should not rely on any user input and hence, be able to produce reconstructions of high quality. To guarantee high quality reconstruction more complex and computationally expensive algorithms are used.

In this thesis we present a pipeline for active, dense, and automatic 3D reconstruction. This pipeline consists of the calibration of the hardware including a new method for consistent calibration of multiple cameras, image acquisition and processing, and last, a novel 3D reconstruction method utilizing the acquired data in the hardware set-up.

State-of-the-art 3D reconstruction combine structured light with photometric stereo reconstruction methods. The novel 3D reconstruction presented in this thesis omits structured light and instead uses multi-view stereo with photometric stereo reconstruction. Using photometric stereo allows to obtain normal information of the object's surface by capturing a sequence of a few images. In contrast to laser triangulation or structured light approaches our approach can capture the object from multiple view points at the same time without any interference. Thus, the acquisition time is independent of the number of viewpoints, which allows to acquire an object from all directions at the same time.

For photometric stereo reconstruction we use a light stage that allows to control the illumination within it. We acquire images of the object under specific illuminations in order to generate normal maps. For multi-view stereo reconstruction we use multiple cameras that acquire images of the object simultaneously from different view points. For correct reconstructions the cameras have to be calibrated with respect to each other and with respect to the light stage. An appropriate calibration method to do this is described. Furthermore, the captured images have to processed accordingly to ensure high reconstruction results. The image development and processing steps as well as the generation of the illumination patterns for the light stage are presented. Finally, the 3D reconstruction method that uses the acquired data in order to generate dense, smooth 3D surfaces of objects is described.

This thesis is structured as follows. In Chap. 2 we give a brief introduction of the background that is used throughout this thesis. This includes a description of the camera model, an overview of optimization methods, and a short overview of existing approaches to 3D reconstruction with related work. Chap. 3 presents our approach to geometric and photometric camera calibration. In Chap. 4 we describe the hardware that we use for acquiring data. In Chap. 5 our method for 3D reconstruction is presented. Last, conclusion and future work is presented in Chap. 6.

Chapter 2

Background

In this chapter we describe the mathematical background and give an overview of available approaches for 3D reconstruction. First, we start with a mathematical description of the camera model that we use for camera calibration. The camera calibration method presented in Chap. 3 is an integral part for the presented pipeline for 3D reconstruction. Next, we give a brief summary of optimization methods that are also used in the presented camera calibration method as well as in the 3D reconstruction method in Chap. 5. Finally, an overview of existing 3D reconstruction techniques is given. This also allows to put the proposed reconstruction method into perspective with alternative methods.

2.1 Camera Model

In this thesis we use the pinhole camera model to describe the underlying process of image formation. It is motivated by the pinhole camera, which is a type of camera without lens. For a mathematical description we introduce homogeneous coordinates before we explain the camera model in detail. Finally, we briefly describe how we model lens distortions that arise from the usage of lenses with a real camera. For more details about the pinhole camera model and the single view geometry that is related to it we refer to [43, 64].

2.1.1 Pinhole Camera

The pinhole camera is a simple camera that allows to describe the process of image formation. With this camera rays originating from the observed scene pass through the tiny pinhole of the camera and are projected to the opposite side of the camera.

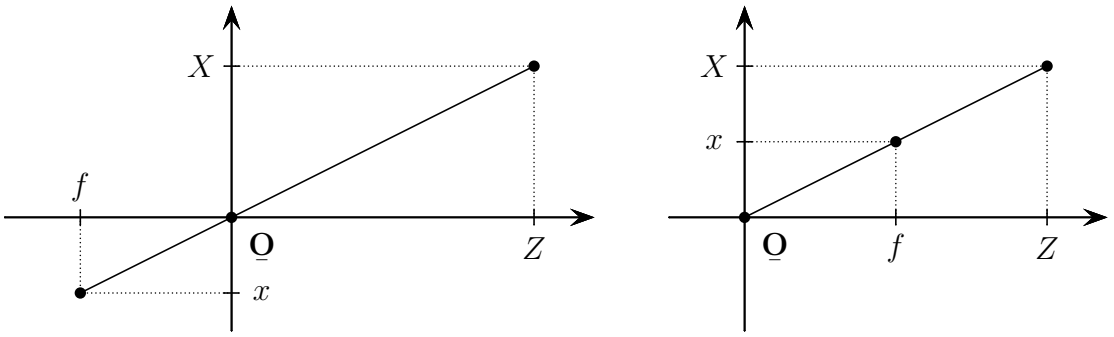

Figure 2.1: Geometry of the pinhole camera model. The pinhole is located in the origin $\underline{\mathbf{O}}$ of the coordinate system. The pinhole camera has focal length f. **Left**: If the image plane is located behind the origin $\underline{\mathbf{O}}$, the image is inverted. **Right**: An upright image is obtained when the (virtual) image plane is located in front of the origin $\underline{\mathbf{O}}$.

Since all rays intersect in the pinhole the image of the scene in the camera is inverted, similar to the human eye where incident rays pass through the pupil and form an inverted image on the retina. According to the theorem of intersecting lines an object of size X at distance Z from the camera appears as an object of size x when projected to the back side of the camera which has a distance of f from the pinhole:

$$\frac{f}{Z} = \frac{x}{X} \tag{2.1}$$

Hence, the size of an object in the formed image x is proportional to the size of the object in the real world X and inversely proportional to the distance Z of the object to the pinhole and the focal length $|f|$, which is the distance between the opposite side of the camera and its pinhole.

Assuming the pinhole of the camera is located in the origin of the coordinate system $\underline{\mathbf{O}}$ and $X, Z > 0$, then $f < 0$ and consequently $x < 0$. Since x is negative the image of the pinhole camera is inverted.

However for the sake of simplicity we assume in our model from now on that the image is formed in front of the pinhole. In this case $f > 0$, consequently x is positive and an upright image is obtained.

The pinhole camera model is a simple model of a real world camera without lens that neither suffers from diffraction phenomena, out-of-focus blur, spherical aberrations, chromatic aberrations, and lens distortions introduced with a camera lens, nor from image noise introduced from CCD/CMOS sensors from digital single lens reflex (DSLR) cameras or film grain from the photographic film in analog cameras.

2.1.2 Projective Space and Homogeneous Coordinates

We will now introduce the concept of projective space and homogeneous coordinates. This allows us to express the projection of a 3D point in world space to a 2D point on the camera sensor in a easy, convenient, and general way.

In this and the next section we use the following notation: We write $\underline{\mathbf{X}}$ to denote a point in the Euclidean space in order to distinguish it from $\mathbf{X}$, which is the same point represented in homogeneous coordinates. A point $\underline{\mathbf{X}} \in \mathbb{R}^n$ in Euclidean space is represented by a line $\mathbf{X} \in \mathbb{R}^{n+1} = \mathbb{P}^n$ in projective space with homogeneous coordinates as follows:

$$\underline{\mathbf{X}} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n \leftrightarrow \mathbf{X} = \begin{pmatrix} w \cdot x_1 \\ \vdots \\ w \cdot x_n \\ w \end{pmatrix} \in \mathbb{P}^n = \mathbb{R}^{n+1}, w \in \mathbb{R}. \tag{2.2}$$

Conversely, any point on the line $\mathbf{X} \neq \mathbf{0} \in \mathbb{P}^n$ represents the same point $\underline{\mathbf{X}} \in \mathbb{R}^n$. Note that we treat points from $\mathbb{R}^n$ given in homogeneous coordinates as if they were points from $\mathbb{R}^{n+1}$. For the sake of simplicity we will transform points form Euclidean space to homogeneous coordinates by

$$\underline{\mathbf{X}} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n \rightarrow \mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} \in \mathbb{P}^n, \tag{2.3}$$

and homogeneous coordinates back to Euclidean space by

$$\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ w \end{pmatrix} \in \mathbb{P}^n \rightarrow \underline{\mathbf{X}} = \begin{pmatrix} \frac{x_1}{w} \\ \vdots \\ \frac{x_n}{w} \end{pmatrix} \in \mathbb{R}^n, \tag{2.4}$$

for $w \neq 0$. If $w = 0$, $\mathbf{X}$ represents a point at infinity in Euclidean space, that may arise for example as a result from computing the intersection of two parallel lines in $\mathbb{P}^2$.

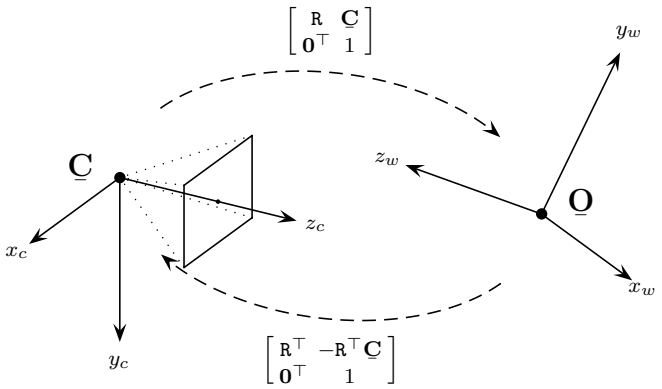

Figure 2.2: Change between the camera coordinate system on the left and the world coordinate system on the right as given by Eqns. 2.5 and 2.6.

2.1.3 Camera Representation

Using homogeneous coordinates the projection of a point $\mathbf{X}_w \in \mathbb{P}^3$ from the world coordinate system to a point $\mathbf{x}_i \in \mathbb{P}^2$ in the image coordinate system of the camera sensor can be described by the following three steps:

1. Transformation of $\mathbf{X}_w \in \mathbb{P}^3$ from the world coordinate system to $\mathbf{X}_c \in \mathbb{P}^3$ in the camera coordinate system.

2. Projection of the point $\mathbf{X}_c \in \mathbb{P}^3$ in 3D space from the camera coordinate system to the 2D point $\mathbf{x}_{ip} \in \mathbb{P}^2$ in the image plane of the camera.

3. Transformation of $\mathbf{x}_{ip} \in \mathbb{P}^2$ from the image plane to $\mathbf{x}_i \in \mathbb{P}^2$ in the image coordinate system, which is the coordinate system of the camera sensor.

In the camera coordinate system the camera center $\underline{\mathbf{C}}$ is located at the origin $\underline{\mathbf{O}}$ and the optical axis of the camera coincides with the z-axis of the coordinate system. Given a point $\mathbf{X}_c$ in the camera coordinate system, the corresponding point $\mathbf{X}_w$ in the world coordinate system can be computed via

$$\mathbf{X}_w = \begin{bmatrix} \mathtt{R} & \underline{\mathbf{C}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_c, \tag{2.5}$$

where $\mathtt{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix describing the orientation of the camera and $\mathbf{0} \in \mathbb{R}^3$ is the null vector. Computing the transformation of a 3D point from the world back to the

camera coordinate system can be done by the inverse transformation of Eqn. 2.5:

$$\mathbf{X}_c = \begin{bmatrix} \mathtt{R}^\top & -\mathtt{R}^\top \underline{\mathbf{C}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_w, \tag{2.6}$$

which can be easily verified, given $\mathtt{R}^{-1} = \mathtt{R}^\top$ and hence $\mathtt{R}^\top\mathtt{R} = \mathtt{R}\mathtt{R}^\top = \mathtt{I}_{3\times3}$, where $\mathtt{I}_{3\times3}$ is the 3×3 identity matrix.

The projection of the 3D point $\mathbf{X}_c$ in the camera coordinate system to the 2D point $\mathbf{x}_{ip}$ in the image plane is computed by Eqn. 2.1. Using homogeneous coordinates this projection can be written as

$$\mathbf{x}_{ip} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X}_c, \tag{2.7}$$

where f is the focal length of the camera.

For the camera position $\underline{\mathbf{C}}$ and the 3D point positions $\mathbf{X}_c$ and $\mathbf{X}_w$ we use metric units. Hence, also the 2D point $\mathbf{x}_{ip}$ representing the projection of the 3D point $\mathbf{X}_w$ onto the image plane is given in metric units. However, the sensor of a camera is an array of cells. Hence for transforming $\mathbf{x}_{ip}$ from the image plane to $\mathbf{x}_i$ in the image coordinate system we information about the geometry and position of the sensor. This information include the pixel size s_x and s_y, the principal point offset p_x and p_y and the skew angle θ. The skew angle θ is the angle between the x- and y-axis of the sensor. The principal point is the intersection of the optical axis or principal axis with the image plane. The principal point offset is the offset of this intersection point in x- and y-direction from the true (geometric) center of the sensor. Finally, the pixel size s_x and s_y is the size of a single sensor cell or *pixel* (picture element) in x- and y-direction. The transformation from the image plane to the image coordinate system is then given by

$$\mathbf{x}_i = \begin{bmatrix} \frac{1}{s_x} & \cot(\theta) & p_x \\ 0 & \frac{\sin(\theta)}{s_y} & p_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{ip}. \tag{2.8}$$

In this representation we assume that the origin of the image coordinate system coincides with the origin of the pixel coordinate system of the image. In this case the obtained 2D point $\mathbf{x}_i$ in the image coordinate system given in homogeneous coordinates is simply

transformed back to the 2D point $\underline{\mathbf{x}}_p$ in the pixel coordinate system:

$$\mathbf{x}_i = \begin{pmatrix} x \\ y \\ w \end{pmatrix} \rightarrow \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \end{pmatrix} = \underline{\mathbf{x}}_p. \tag{2.9}$$

However, usually the origin of the pixel coordinate system in a digital image is located in the top left corner of the image. In this case $\mathbf{x}_p$ is obtained by back-transformation to Euclidean space and translation:

$$\mathbf{x}_i = \begin{pmatrix} x \\ y \\ w \end{pmatrix} \rightarrow \begin{pmatrix} \frac{x}{w} + \frac{width - 1}{2} \\ \frac{y}{w} + \frac{height - 1}{2} \end{pmatrix} = \underline{\mathbf{x}}_p, \tag{2.10}$$

where *width* and *height* are the width and height of the image. From now on we will not mention this translation explicitly and assume that it will be always applied correctly during the back-transformation from homogeneous coordinates to Euclidean coordinates.

Summing up the above steps a 3D point $\mathbf{X}_w$ in world coordinates is projected to a 2D point $\mathbf{x}_i$ in image coordinates by:

$$\mathbf{x}_i = \begin{bmatrix} \frac{1}{s_x} & \cot(\theta) & p_x \\ 0 & \frac{\sin(\theta)}{s_y} & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R}^\top & -\mathtt{R}^\top \underline{\mathbf{C}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_w. \tag{2.11}$$

We can rewrite this equation to the projection equation

$$\mathbf{x}_i = \mathtt{K} \begin{bmatrix} \mathtt{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathtt{R}^\top & -\mathtt{R}^\top \underline{\mathbf{C}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_w, \tag{2.12}$$

where $\mathtt{R}$ is a 3×3 rotation matrix describing the orientation of the camera, $\underline{\mathbf{C}}$ is the camera center, and $\mathtt{K}$ is the so called the calibration matrix. The orientation and camera center are called *extrinsic camera parameters*. The focal length f, the pixel size s_x and s_y, the principal point offset p_x and p_y, and the skew angle θ are called *intrinsic camera parameters*.

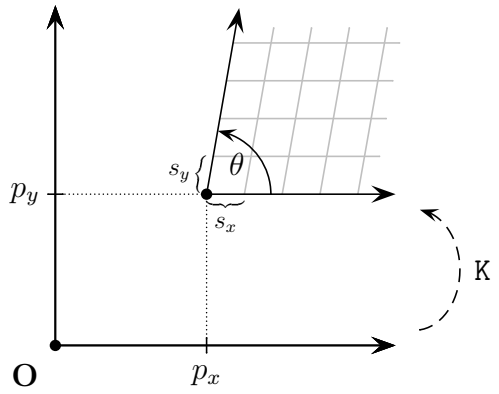

Figure 2.3: Change from the coordinate system of the image plane to the pixel coordinate system by the calibration matrix K.

Those parameters are part of the calibration matrix K:

$$\mathtt{K} = \begin{bmatrix} \frac{f}{s_x} & \cot(\theta) & p_x \\ 0 & \frac{f}{s_y}\sin(\theta) & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.13}$$

As shown in Fig. 2.3, the calibration performs a change from the coordinate system of the image plane to the pixel coordinate system. Usually the pixel size s_x and s_y are given by the camera manufacturer or can be computed given the physical sensor size and the image size of an acquired image in pixels. From the calibration matrix we can see, that only the ratios $\frac{f}{s_x}$ and $\frac{f}{s_y}$ are necessary to uniquely determine K, although there are three parameters: the focal length f and the pixel size s_x and s_y. Alternatively we can also replace $\frac{f}{s_x}$ by f and $\frac{f}{s_y}$ by $\frac{fs_x}{s_y} = \alpha f$, where α is the so called *aspect ratio* $\frac{s_x}{s_y}$ resulting in two unknowns f and α. Furthermore, we can see that K is an upper triangular matrix.

The projection equation Eqn. 2.12 can be written in a short way using the 3×4 projection matrix P:

$$\mathbf{x}_i = \mathtt{P} \cdot \mathbf{X}_w, \qquad \text{with } \mathtt{P} = \begin{bmatrix} \mathtt{p}_{11} & \mathtt{p}_{12} & \mathtt{p}_{13} & \mathtt{p}_{14} \\ \mathtt{p}_{21} & \mathtt{p}_{22} & \mathtt{p}_{23} & \mathtt{p}_{24} \\ \mathtt{p}_{31} & \mathtt{p}_{32} & \mathtt{p}_{33} & \mathtt{p}_{34} \end{bmatrix}. \tag{2.14}$$

Every camera matrix of the form given in Eqn. 2.12 can be written in the form of Eqn. 2.14. Such a camera is called *projective camera*. The converse however, is not true. We split the

camera matrix $\mathtt{P}$ as $\mathtt{P} = \begin{bmatrix}\mathtt{B}\ \mathbf{b}\end{bmatrix}$ into a left submatrix $\mathtt{B} \in \mathbb{R}^{3\times3}$ and a vector $\mathbf{b} \in \mathbb{R}^3$. If $\mathtt{B}$ can be decomposed as

$$\mathtt{B} = \mathtt{K} \cdot \mathtt{R}, \tag{2.15}$$

with a upper-triangular calibration matrix $\mathtt{K}$ as given above and a 3×3 rotation matrix $\mathtt{R}$, then the matrix is called *metric camera*. This decomposition of the matrix $\mathtt{B}$ can be done by RQ decomposition which is similar to the QR decomposition given in 2.2.1. Another special case is the *orthographic camera*.

The projection matrix has 12 entries, however, it has only 11 degrees of freedom. This is because $\mathtt{P}$ is defined up to scale: $\mathtt{P}$ represents the same projection matrix as $s\mathtt{P}$ with $s \neq 0$ in the same way as $\mathbf{X}$ represents the same point as $s\mathbf{X}$ with $s \neq 0$ in Euclidean space, which can be verified by Eqn. 2.4. The 11 degrees of freedom correspond to the 5 intrinsic and 6 extrinsic camera parameters.

Rotations in 3D space only have 3 degrees of freedom. Hence, the 3×3 rotation matrix $\mathtt{R}$ in Eqn. 2.12 only has 3 degrees of freedom although $\mathtt{R}$ has 9 entries. There are different ways to describe this matrix. Using a parametrization allows to easily enforce the matrix to be a rotation matrix. Different ways how to parametrize exist, which will be described shortly in the following:

Euler Angles

Any rotation in 3D space can be expressed as a composition of three rotations around three coordinate axes. The three rotations can be carried out be performing three rotations around the axes of the original (fixed) coordinate system or around the axes of a coordinate system obtained from the previous rotations. The rotations are called *extrinsic rotations* and *intrinsic rotations*, respectively.

The order of the three rotations is given by the order of the rotation axes. For example, a rotation around the x-axis, followed by rotations around the original y- and z-axis, is denoted as x-y-z. The same order of rotations in a rotating coordinate system is denoted by x-y'-z''.

Rotations where the first and third rotation is performed around the same axis are also called *proper Euler angles* in order to distinguish them from the so called *Tait-Bryan angles*.

Throughout the rest of this thesis we will use the y-x'-z'' convention for the Euler angles. The rotation angles around the y-, x-, and z-axis we will call pan angle φ, tilt angle ϑ, and roll angle ρ (see Fig. 2.4).

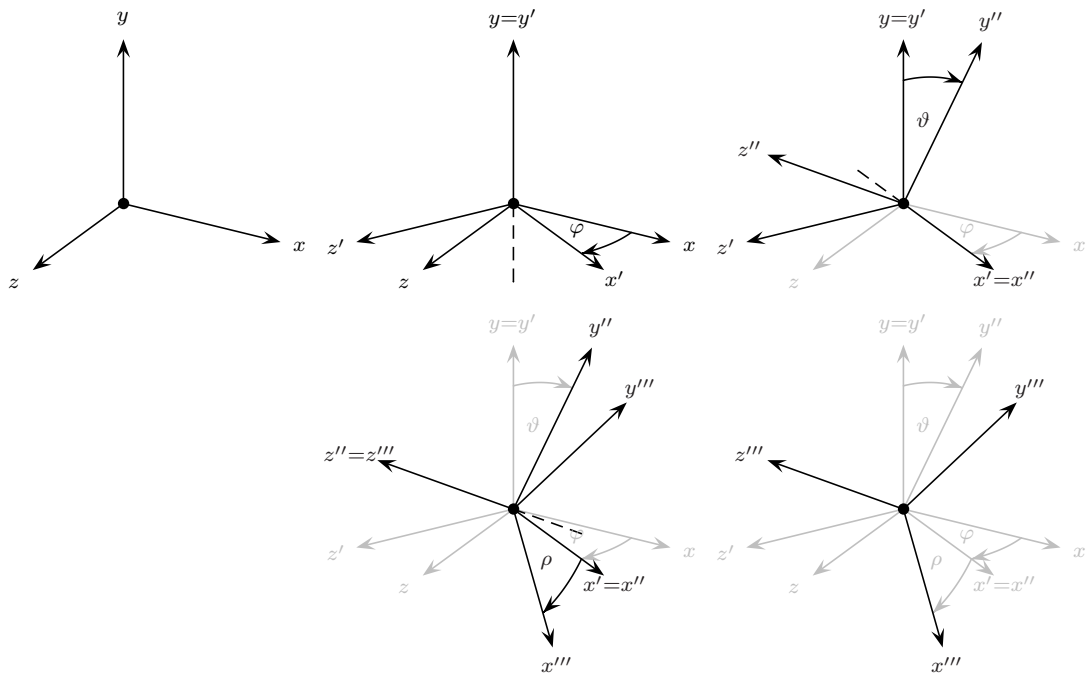

Figure 2.4: Example of a rotation in 3D using the three Euler angles φ (pan), ϑ (tilt), and ρ (roll) according to the y-x'-z'' convention as used in this thesis. From **top left** to **bottom right**: World coordinate system, pan around the y-axis, tilt around the rotated x-axis, roll around the rotated z-axis, local coordinate system after rotation.

The rotation matrix $\mathtt{R}$ can be written as

$$\mathtt{R} = \mathtt{R}_z(\rho) \cdot \mathtt{R}_x(\vartheta) \cdot \mathtt{R}_y(\varphi), \tag{2.16}$$

with parameters φ, ϑ, and ρ, where

$$\begin{aligned}
\mathtt{R}_z(\rho) &= \begin{bmatrix} \cos(\rho) & -\sin(\rho) & 0 \\ \sin(\rho) & \cos(\rho) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\
\mathtt{R}_x(\vartheta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\vartheta) & -\sin(\vartheta) \\ 0 & \sin(\vartheta) & \cos(\vartheta) \end{bmatrix}, \\
\mathtt{R}_y(\varphi) &= \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix}.
\end{aligned} \tag{2.17}$$

Explicitly written we get:

$$\mathtt{R} = \begin{bmatrix} \cos(\rho) & -\sin(\rho) & 0 \\ \sin(\rho) & \cos(\rho) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\vartheta) & -\sin(\vartheta) \\ 0 & \sin(\vartheta) & \cos(\vartheta) \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} = \begin{bmatrix} \mathtt{r}_{11} & \mathtt{r}_{12} & \mathtt{r}_{13} \\ \mathtt{r}_{21} & \mathtt{r}_{22} & \mathtt{r}_{23} \\ \mathtt{r}_{31} & \mathtt{r}_{32} & \mathtt{r}_{33} \end{bmatrix}, \tag{2.18}$$

with

$$\begin{aligned} \mathtt{r}_{11} &= \cos(\rho)\cos(\varphi) - \sin(\rho)\sin(\vartheta)\sin(\varphi), \\ \mathtt{r}_{12} &= -\sin(\rho)\cos(\vartheta), \\ \mathtt{r}_{13} &= \cos(\rho)\sin(\varphi) + \sin(\rho)\sin(\vartheta)\cos(\varphi), \\ \mathtt{r}_{21} &= \sin(\rho)\cos(\varphi) + \cos(\rho)\sin(\vartheta)\sin(\varphi), \\ \mathtt{r}_{22} &= \cos(\rho)\cos(\vartheta), \\ \mathtt{r}_{23} &= \sin(\rho)\sin(\varphi) - \cos(\rho)\sin(\vartheta)\cos(\varphi), \\ \mathtt{r}_{31} &= -\cos(\vartheta)\sin(\varphi), \\ \mathtt{r}_{32} &= \sin(\vartheta), \text{and} \\ \mathtt{r}_{33} &= -\cos(\vartheta)\cos(\varphi). \end{aligned}$$

One disadvantage of the parametrization of rotations with Euler angles is the so called *Gimbal lock*. The Gimbal lock describes the loss of one degree of freedom of the rotation matrix. Given Eqn. 2.18 this is the case for $\vartheta = \frac{\pi}{2}$.

Rotations in 3D can also be modeled using quaternions, an extension of the complex numbers, or using Rodrigues' rotation formula, which make the rotation axis $\mathbf{V}$ and rotation angle α explicit. Here, we will not go into further detail.

2.1.4 Lens Distortion

The pinhole camera is the simplest camera model, where the light from the scene travels through a tiny pinhole and illuminates the back of the camera. While in theory this model can describe the process of image formation, it has disadvantages that makes it usage impractical. Since the pinhole is tiny, the amount of light passing through it, is very small. In order to capture an image the exposure time, that is the time the light is reaching the photographic film or camera sensor, has to be very long. A long exposure time can only be used, if the imaged scene is static, otherwise moving objects will introduce motion blur.

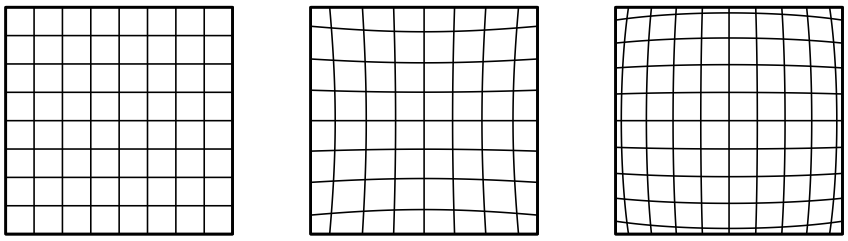

Figure 2.5: Examples of radial lens distortion acting on the image of a grid. The radial lens distortion parameter k_1 is varied, while all other parameters k_i, for $i > 1$ are 0. **Left**: Undistorted image of grid, parallel lines appear parallel to each other, $k_1 = 0$. **Middle**: *Pincushion distortion*, straight lines appear to be bend towards the image center, their outline resemble the shape of a pincushion, $k_1 > 0$. **Right**: *Barrel distortion*, straight lines appear to bend away from the image center, their outline resemble the shape of a barrel, $k_1 < 0$.

Using a more sensitive photographic film or increasing the sensitivity of the camera sensor will introduce grain noise or photon noise, respectively.

Instead a real camera use a camera lens, which is an arrangement of individual lenses, that are used to focus the incoming light to the focal plane at distance f. Because more light can pass through the lens the exposure time can be significantly reduced. However objects that are too close or too far away from the focus range of the camera lens are blurred. The focus range is also called depth of field and indicates the depth from the camera in which objects are imaged sharply.

Because a camera lens consists of several lenses that have imprecisions because of the manufacturing process and each single lens only approximate a theoretical lens in its properties the camera lens introduces so-called *lens distortions* [108, 24].

One can distinguish two kinds of lens distortion. With radial lens distortion a point in the image plane is moved in radial direction from the principal point only. For simple examples of the effects of radial lens distortion, see Fig. 2.5. This kind of distortion usually occurs because each individual lens in the camera lens has a varying thickness and hence different refraction properties depending on the distance from the lens center. With tangential lens distortion on the other hand, a point in the image plane is moved in the tangential direction compared to radial lens distortion. This kind of distortion usually comes from slight imprecision in the alignment of the individual lenses. Radial lens distortion is by far more dominant in acquired images than tangential lens distortion [161]. Only if the camera has to be calibrated with very high accuracy tangential lens distortion has to be considered.

Given the undistorted 2D point $\mathbf{x}_u = (x_u, y_u)^\top$ of a 3D point $\mathbf{X}$ projected onto the image the distorted 2D point $\mathbf{x}_d = (x_d, y_d)^\top$ is modeled in the case of radial lens distortion only as

$$x_d = x_u\Big(1 + k_1 r^2 + k_2 r^4 + \dots\Big), \tag{2.19}$$

$$y_d = y_u\Big(1 + k_1 r^2 + k_2 r^4 + \dots\Big), \tag{2.20}$$

and in the case of radial and tangential lens distortion as

$$\begin{aligned} x_d = {} & x_u\Big(1 + k_1 r^2 + k_2 r^4 + \dots\Big) \\ & + \Big(p_2(r^2 + 2x_u^2) + 2p_1 x_u y_u\Big)\Big(1 + p_3 r^2 + p_4 r^4 + \dots\Big), \end{aligned} \tag{2.21}$$

$$\begin{aligned} y_d = {} & y_u\Big(1 + k_1 r^2 + k_2 r^4 + \dots\Big) \\ & + \Big(p_1(r^2 + 2y_u^2) + 2p_2 x_u y_u\Big)\Big(1 + p_3 r^2 + p_4 r^4 + \dots\Big). \end{aligned} \tag{2.22}$$

Here $r = \sqrt{x_u^2 + y_u^2}$ is the distance of $\mathbf{x}_u$ to the principal point in the image plane, $k_1, k_2, \dots$ are the radial distortion coefficients, and $p_1, p_2, p_3, p_4, \dots$ are the tangential distortion coefficients [24, 25].

Introducing lens distortion to Eqn. 2.11 results in the following three steps, since the higher order terms in Eqns. 2.19 – 2.22 cannot be expressed in a compact matrix notation:

1. Computing the undistorted projection $\mathbf{x}_{ip,u}$ of the 3D point $\mathbf{X}_w$ according to Eqn. 2.6 and 2.7:

$$\mathbf{x}_{ip,u} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R}^\top & -\mathtt{R}^\top \mathbf{C} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_w. \tag{2.23}$$

2. Computing the distorted point $\mathbf{x}_{ip,d}$ in the image plane from the undistorted one ($\mathbf{x}_{ip,u}$) according to Eqns. 2.19 and 2.20 or Eqns. 2.21 and 2.22, respectively.

3. Computing the pixel position $\mathbf{x}_x$ given the position of the distorted projection $\mathbf{x}_{ip,d}$ of $\mathbf{X}_w$ according to Eqn. 2.8

$$\mathbf{x}_i = \begin{bmatrix} \frac{1}{s_x} & \cot(\theta) & p_x \\ 0 & \frac{\sin(\theta)}{s_y} & p_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{ip,d}. \tag{2.24}$$

As mentioned above, in practice radial lens distortion dominates tangential lens distortion. Hence estimating the radial lens distortion with distortion coefficients k_1 and k_2 is usually sufficient to for a accurate camera calibration. Note that for larger and qualitatively better camera lenses estimating k_1 can also be sufficient since less distortion is present, while for smaller camera lenses, for example in mobile phones, significantly more distortion is present and it is therefore advisable to use more coefficients in the estimation process. Lens distortion parameters are intrinsic camera parameters.

2.1.5 Camera Calibration Methods

The process of estimating the intrinsic and (relative) extrinsic camera parameters in Eqns. 2.11 and 2.19 – 2.22 is called camera parameter estimation. Once camera parameter have been estimated or if the camera parameters are already known the camera is said to be calibrated. Hence this process is also called camera calibration.

Depending on the setting and the available data different parameters can be estimated. From a sequence of images of a static scene (or static parts of a scene) captured by a single moving camera the intrinsic camera parameter and the motion of the camera can be estimated up to a scaling factor. This process is also known as camera auto-calibration. In this context no information about the scene itself is known.

In classic camera calibration methods usually a calibration object, calibration device, or calibration pattern is used. In this context the geometry of the object is known such that intrinsic and extrinsic camera parameters can be estimated from a single image. Here, the position and orientation is estimated with respect to the calibration object. The calibration object is usually three-dimensional, however there also exists methods that can estimate parameters from a planar calibration object. Given additional scene information, such as the positions of lines in one image that are parallel in the real world, auto-calibration can also be performed with one image only.

General Approach

Classic camera calibration methods use a calibration object with known geometry [67, 20, 53]. Such methods are also known as *chart based calibration* methods. For estimating the five intrinsic camera parameters focal length f, principal point offset p_x, p_y, skew angle θ, and aspect ratio α as well as the six extrinsic camera parameters (three for position and three for orientation) correspondences between 2D feature points $\mathbf{x}_i$ in the image and 3D

points $\mathbf{X}_i$ of the calibration object have to be established. The 3D points $\mathbf{X}_i$ are known, only the corresponding 2D points in the image have to be detected. In the context of camera calibration the problem of estimating the extrinsic parameters with known intrinsic camera parameters is also called *pose estimation* [76, 90, 4].

Typically two-dimensional calibration object with black-white features are used. Such patterns may be either a checkerboard pattern, or patterns consisting of squares or circles arranged in a regular grid or even irregularly. Alternatively, a planar pattern is moved towards or away from the camera in a precisely defined way to generate samples points in all three dimensions. Using a black-white pattern simplifies the task of extracting the 2D feature points $\mathbf{x}_i$ of the pattern by segmentation methods, such as thresholding. Once the black regions of the calibration object have been detected either their centers or (in case of rectangles or squares) their four corner points are computed. While computing the centers is easier and faster, computing the four corner points of each square as intersection of its four edges gives four times more points that can be used for parameter estimation, but it also involves more sophisticated algorithms.

Once the 2D-3D correspondences $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ have been established, camera parameters can be estimated. This task usually involves the solving of a linear system of equations, which can be achieved for example by using a *singular value decomposition* (SVD, 2.33). Additionally, the estimated camera parameters can be improved by minimizing the projection error

$$\sum_i d\Big(\tilde{\mathbf{x}}_i, \hat{\mathtt{P}}\bar{\mathbf{X}}_i\Big)^2. \tag{2.25}$$

Here, $\tilde{\mathbf{x}}_i$ is the measured 2D point corresponding to the known 3D point $\bar{\mathbf{X}}_i$ of the calibration pattern, $\hat{\mathtt{P}}\bar{\mathbf{X}}_i$ is the projection of the 3D point into the image plane using the estimated projection matrix $\hat{\mathtt{P}}$, and $d(\mathbf{a}, \mathbf{b}) = \sqrt{||\mathbf{a} - \mathbf{b}||}$ is the Euclidean distance. A method for minimizing the projection error is presented in Sec. 2.2.4.

In the case of lens distortion a linear estimation of the camera parameters is performed first, neglecting lens distortion. Then, lens distortion parameters are introduced in the process of minimizing the projection error as described in Eqn. 2.25 using Eqns. 2.19 – 2.24.

Such an approach to camera parameter estimation can be always used when information about the 3D geometry of the calibration object is available. However this approach does not work, if the 3D points $\mathbf{X}_i$ are coplanar (for example in the case of $Z_i = 0$). Then, the system matrix used in the initial parameter estimation will not have full column rank. Hence, not all camera parameters of the projection matrix Eqn. 2.14 can be estimated in

this case.

Tsai's Method

One possibility to deal with such cases is to use Tsai's approach [161]. It is a two stage approach that uses a planar calibration pattern to estimate the parameters of

$$\mathbf{x}_i = \begin{bmatrix} \frac{f}{s_x} & s & 0 & 0 \\ 0 & \frac{f}{s_y} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R}^\top & -\mathtt{R}^\top \mathbf{C} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_i, \tag{2.26}$$

in the presence of radial lens distortion of the form $x_d = x_u + x_u k_1 r^2$, $y_d = y_u + y_u k_1 r^2$ given correspondences $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ and pixel sizes s_x, s_y. Here, we have the skew parameter s instead of the angle of skew θ. In both stages different sets of camera parameters are estimated in order to reduce the complexity of the estimation problem. In the first stage the orientation $\mathtt{R}$ of the camera and the position $(\mathbf{C}_x, \mathbf{C}_y)^\top$ parallel to the calibration pattern is computed exploiting geometric constraints. In the second stage the focal length f, radial lens distortion k_1 and the distance $\mathbf{C}_z$ from the calibration pattern is computed. This is done by computing f and $\mathbf{C}_z$ assuming $k_1 = 0$ as initial solution for the following optimization of f, $\mathbf{C}_z$, and k_1.

While this method allows to estimate camera parameters using a single input image of the planar calibration pattern, care has to be taken in order to avoid positioning the camera parallel to the calibration pattern. In this specific case the focal length and distance from the calibration pattern cannot be estimated reliably at the same time anymore. This is because there is no variation in the depth of the feature points of the calibration pattern in the acquired image. We can see from Eqn. 2.7 with $\mathbf{X}_c = (0, 0, X_z, 1)^\top$ that changing the distance X_z of a feature point from the camera can be exactly compensated by adjusting the focal length f appropriately.

Zhang's Method

The method of Zhang [174] is another approach that can estimate camera parameters from a planar calibration pattern. However, the planar calibration pattern has to be viewed in at least two different orientations. In this approach the projection matrix $\mathtt{P}$ is estimated without lens distortion parameters.Then, the projection matrix is split into a submatrix

$\mathtt{B} \in \mathbb{R}^{3\times3}$ and a vector $\mathbf{b} \in \mathbb{R}^3$: $\mathtt{P} = \begin{bmatrix}\mathtt{B}\ \mathbf{b}\end{bmatrix}$. Constraints on the image of the absolute conic and its image are used to recover the calibration matrix $\mathtt{K}$ from $\mathtt{B}$.

The absolute conic and its image is a entity from projective geometry. The image of the absolute conic is a set of purely imaginary points that neither depend on the position nor on the orientation of the camera. This is comparable to vanishing points and vanishing lines. A vanishing point is a purely imaginary point. It is the intersection of parallel lines in the real world. One can think of parallel lines intersecting at infinity and the image of the vanishing point being the projection of that intersection point at infinity. Pairs of parallel lines in the real world can have different directions. This means that there are different vanishing points depending on the direction of the parallel lines. The set of all vanishing points, which is the set of vanishing points belonging to parallel lines of all directions, is called the vanishing line. Because the vanishing line contains vanishing points that are all located at infinity the position of the camera does not change the position of the vanishing points in the image and hence also does not change the position of the vanishing line in the image.

In the same way the image of the absolute conic is independent on the extrinsic camera parameters. Hence it can be used to estimate the intrinsic camera parameters. With the intrinsic parameters estimated the extrinsic parameters are computed from $\mathtt{K}$ and $\mathbf{b}$. Finally the camera parameters are optimized by minimizing the projection error from Eqn. 2.25 under lens distortion.

Auto-calibration

In contrast to the methods described in the previous sections, one or several cameras can also be calibrated without a calibration object with known geometry. Such a calibration method is called auto-calibration or self-calibration. In auto-calibration cameras are calibrated from a set of uncalibrated images using constraints on the intrinsic camera parameters and a so-called *projective reconstruction*. A more detailed description of the projective reconstruction and the auto-calibration process is given in Sec. 2.3.1.

2.2 Optimization

Minimizing cost functions, for example the projection error in Eqn. 2.25, or optimizing parameters, as for example in camera parameter estimation, requires optimization methods. One practically important class of optimization methods are iterative optimization meth-

ods. Those methods do not compute the optimal solution directly. Instead, by using an appropriate approximation they iteratively refine an initial solution to converge towards the optimal solution. Directly computing the optimal solution is desirable, however for complex problems that arise analytic solutions typically do not exist. Hence, iterative methods are used for optimization. In this section we first introduce the problem, briefly present some optimization methods for least squares problems [95, 49, 122, 129], and finally describe the random sampling consensus method [47].

In this context we are given an error function $f : \mathbb{R}^n \to \mathbb{R}^m$. We want to optimize a low dimensional estimation parameter vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ given f in such a way that the function values $f(\hat{\mathbf{x}})$ resembles a high dimensional measurement vector $\tilde{\mathbf{b}} \in \mathbb{R}^m$ with $m > n$:

$$f(\hat{\mathbf{x}}) = \tilde{\mathbf{b}}. \tag{2.27}$$

The function f is representing some error or cost function. The measurement vector $\tilde{\mathbf{b}}$ is usually higher dimensional than the parameter vector $\hat{\mathbf{x}}$ and since the measuring itself is not perfect small errors are also introduced to the measured data. Because of those errors and the high dimensionality the measurements are contradicting each other. Hence, a solution to Eqn. 2.27 does not exist. Instead, one can compute the parameters that minimize the resulting error $||\boldsymbol{\varepsilon}|| = \left|\left|f(\hat{\mathbf{x}}) - \tilde{\mathbf{b}}\right|\right|$. That means given the function f, and the measurement $\tilde{\mathbf{b}}$ compute

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left|\left|f(\mathbf{x}) - \tilde{\mathbf{b}}\right|\right|^2. \tag{2.28}$$

Estimating $\hat{\mathbf{x}}$ in such a way is equivalent to finding the parameters $\hat{\mathbf{x}}$ that best describe the measured data $\tilde{\mathbf{b}}$ in a given model represented here by the function f.

2.2.1 Linear Least Squares

Computing Eqn. 2.28 in the case of the function f being linear is a simple optimization problem. This leads to the method of linear least squares. In this case f can be simply represented by a real-valued matrix $\mathtt{A} \in \mathbb{R}^{m \times n}$: $f(\hat{\mathbf{x}}) = \mathtt{A}\hat{\mathbf{x}}$. Furthermore, Eqn. 2.27 simplifies to

$$\mathtt{A}\hat{\mathbf{x}} = \tilde{\mathbf{b}} \tag{2.29}$$

and we obtain a linear system of equations. In the presence of measurement errors and because of the measurement vector $\tilde{\mathbf{b}} \in \mathbb{R}^m$ having a higher dimension than the parameter vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ this linear system of equations is overdetermined and a solution usually does

not exists. We obtain the *normal equation* by multiplying both sides with $\mathtt{A}^\top$:

$$\mathtt{A}^\top \mathtt{A}\hat{\mathbf{x}} = \mathtt{A}^\top \tilde{\mathbf{b}} \tag{2.30}$$

By construction the resulting *normal matrix* $\mathtt{C} = \mathtt{A}^\top \mathtt{A}, \mathtt{C} \in \mathbb{R}^{n \times n}$ on the left hand side is a symmetric, positive semi-definite matrix. Also, the rank of $\mathtt{C}$ is equal to the rank of $\mathtt{A}$. If $\mathtt{A}$ has full rank, then $\mathtt{C}$ also has full rank and therefore $\mathtt{C}$ is positive definite and invertible. In this case the linear system of equations

$$\mathtt{C}\hat{\mathbf{x}} = \mathbf{d}, \tag{2.31}$$

with $\mathbf{d} = \mathtt{A}^\top \tilde{\mathbf{b}}, \mathbf{d} \in \mathbb{R}^n$ can be solved for $\hat{\mathbf{x}}$. This system can be solved by inverting $\mathtt{C}$. The solution is then given by $\hat{\mathbf{x}} = \mathtt{C}^{-1}\mathbf{d}$. The easiest way to compute $\mathtt{C}^{-1}$ is to use the method of Gaussian elimination. However, there are better ways to solve this equation for $\hat{\mathbf{x}}$ in terms of computation time, accuracy, and numerical stability than by directly computing the inverse.

Matrix Decompositions

Different methods exist that solve linear systems of equations as given in Eqns. 2.29 and 2.31. Those methods decompose the system matrices into several easier to invert matrices. Common decompositions will be explained briefly:

1. The *eigenvalue decomposition* or *eigendecomposition* of $\mathtt{C} \in \mathbb{R}^{n \times n}$ is given by

$$\mathtt{C} = \mathtt{Q} \cdot \mathtt{E} \cdot \mathtt{Q}^\top, \tag{2.32}$$

   with $\mathtt{Q} \in \mathbb{R}^{n \times n}$ being an orthogonal matrix such that $\mathtt{Q}^\top \cdot \mathtt{Q} = \mathtt{I}_{n \times n}$ and $\mathtt{E} \in \mathbb{R}^{n \times n}$ being a diagonal matrix with the *eigenvalues* of $\mathtt{C}$ on the main diagonal. Furthermore the ith eigenvalue in $\mathtt{E}$ corresponds to the ith (column) vector of $\mathtt{Q}$, which is the *eigenvector*. For a real-valued matrix $\mathtt{C}$ a unique eigendecomposition always exists. Given such a decomposition Eqn. 2.30 can be solved in three steps. First, solve $\mathtt{Q}\mathbf{z} = \mathbf{d}$ for $\mathbf{d}$ by computing $\mathbf{z} = \mathtt{Q}^\top \mathbf{d}$. Then, solve $\mathtt{E}\mathbf{y} = \mathbf{z}$ for $\mathbf{y}$ by component-wise division of the entries of $\mathbf{z}$ by the corresponding eigenvalues in $\mathtt{D}$. Finally, solve $\mathtt{Q}^\top \mathbf{x} = \mathbf{y}$ for $\mathbf{x}$ by computing $\mathbf{x} = \mathtt{Q}\mathbf{y}$. The eigendecomposition is also known as *principal component analysis* (PCA).

2. The *singular value decomposition* is a generalization of the eigendecomposition 2.32 to non-square matrices. The real-valued matrix of the overdetermined linear system of equations $\mathtt{A} \in \mathbb{R}^{m \times n}$ given in Eqn. 2.29 can be decomposed to

$$\mathtt{A} = \mathtt{U} \cdot \mathtt{S} \cdot \mathtt{V}^\top, \tag{2.33}$$

where $\mathtt{U} \in \mathbb{R}^{n \times n}$ and $\mathtt{V} \in \mathbb{R}^{m \times m}$ are orthogonal matrices with $\mathtt{U}^\top \cdot \mathtt{U} = \mathtt{I}_{n \times n}$, $\mathtt{V}^\top \cdot \mathtt{V} = \mathtt{I}_{m \times m}$ and $\mathtt{S} \in \mathbb{R}^{m \times n}$ is a diagonal matrix with the *singular values* of $\mathtt{A}$ on the diagonal. The singular values of $\mathtt{A}$ are the square roots of the eigenvalues of $\mathtt{A}^\top \cdot \mathtt{A}$. Similar to the eigendecomposition the ith (column) vector of $\mathtt{U}$ corresponding to the ith singular value is called the left singular vector and the ith (column) vector of $\mathtt{V}$ corresponding to the ith singular value is called the right singular vector. The left singular vectors are the eigenvectors of $\mathtt{A} \cdot \mathtt{A}^\top$ and the right singular vectors are the eigenvectors of $\mathtt{A}^\top \cdot \mathtt{A} = \mathtt{C}$. Usually the singular values are order in decreasing order. Then, the singular value decomposition of matrix $\mathtt{A}$ is unique. Given such a decomposition a solution to Eqn. 2.29 can be computed in the following three steps. First, solve $\mathtt{U}\mathbf{z} = \mathbf{b}$ by computing $\mathbf{z}$ as $\mathbf{z} = \mathtt{U}^\top\mathbf{b}$. Second solve $\mathtt{S}\mathbf{y} = \mathbf{z}$ for $\mathbf{y}$ by multiplying $\mathbf{z}$ component-wise with the inverse of the singular values. Last, solve $\mathtt{V}^\top\mathbf{x} = \mathbf{y}$ by computing $\mathbf{x}$ as $\mathbf{x} = \mathtt{V}\mathbf{y}$.

3. The *QR decomposition* is inspired by the method of Gram-Schmidt for the orthogonalization of a set of vectors. It computes a decomposition of $\mathtt{C} \in \mathbb{R}^{n \times n}$ as

$$\mathtt{C} = \mathtt{Q} \cdot \mathtt{R}, \tag{2.34}$$

where $\mathtt{Q} \in \mathbb{R}^{n \times n}$ is a orthogonal matrix such that $\mathtt{Q}^\top \cdot \mathtt{Q} = \mathtt{I}_{n \times n}$ and $\mathtt{R} \in \mathbb{R}^{n \times n}$ is an upper triangular matrix. For an invertible matrix $\mathtt{C}$ and matrix $\mathtt{R}$ having positive entries on the main diagonal this decomposition is unique. With a decomposition of $\mathtt{C}$ a solution for Eqn. 2.30 is obtained as follows. First, solve the linear system of equations $\mathtt{Q}\mathbf{y} = \mathbf{d}$ for $\mathbf{y}$ which is equivalent to computing $\mathbf{y} = \mathtt{Q}^\top\mathbf{d}$. Then, solve $\mathtt{R}\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$ by *back substitution.*

4. The *LU decomposition* computes

$$\mathtt{C} = \mathtt{L} \cdot \mathtt{U}, \tag{2.35}$$

where $\mathtt{C} \in \mathbb{R}^{n \times n}$ is decomposed into a lower triangular matrix $\mathtt{L} \in \mathbb{R}^{n \times n}$ and an upper triangular matrix $\mathtt{U} \in \mathbb{R}^{n \times n}$. This decomposition exists and is unique, if and only if $\mathtt{C}$

is invertible. Having the decomposition $\mathtt{C} = \mathtt{L} \cdot \mathtt{U}$ we solve Eqn. 2.30 in the following way. The linear system of equations $\mathtt{L}\mathbf{y} = \mathbf{d}$ can be efficiently solved by *forward substitution* for $\mathbf{y}$, since $\mathtt{L}$ is an lower triangular matrix. Finally, $\mathtt{U}\mathbf{x} = \mathbf{y}$ is efficiently solved by back substitution for $\mathbf{x}$ with $\mathtt{U}$ being an upper triangular matrix.

5. The *Cholesky decomposition* for the real-valued matrix $\mathtt{C} \in \mathbb{R}^{n \times n}$ computes

$$\mathtt{C} = \mathtt{L} \cdot \mathtt{L}^\top, \tag{2.36}$$

with $\mathtt{L} \in \mathbb{R}^{n \times n}$ being a lower triangular matrix. Every real-valued invertible matrix $\mathtt{C}$ can be uniquely decomposed in such a way. It follows from Eqn. 2.35 by exploiting the fact that $\mathtt{C}$ is symmetric. Similarly to the LU decomposition, Eqn. 2.30 is solved by first solving the linear system of equations $\mathtt{L}\mathbf{y} = \mathbf{d}$ by forward substitution and then solving $\mathtt{L}^\top\mathbf{x} = \mathbf{y}$ by back substitution.

6. The *LDL decomposition* is based on the Cholesky decomposition. Let $\mathtt{L} \in \mathbb{R}^{n \times n}$ be a lower triangular matrix with all main diagonal entries being 1 and $\mathtt{D} \in \mathbb{R}^{n \times n}$ be a diagonal matrix. With $\mathtt{L}' = \mathtt{L} \cdot \mathtt{D}^{\frac{1}{2}}$, $\mathtt{L}' \in \mathbb{R}^{n \times n}$ from Eqn. 2.36 we obtain $\mathtt{C} = \mathtt{L}' \cdot \mathtt{L}'^\top = \mathtt{L} \cdot \mathtt{D}^{\frac{1}{2}} \cdot \left(\mathtt{L} \cdot \mathtt{D}^{\frac{1}{2}}\right)^\top = \mathtt{L} \cdot \mathtt{D}^{\frac{1}{2}} \cdot \mathtt{D}^{\frac{1}{2}} \cdot \mathtt{L}^\top$, which is

$$\mathtt{C} = \mathtt{L} \cdot \mathtt{D} \cdot \mathtt{L}^\top. \tag{2.37}$$

It is also related to the eigendecomposition 2.32. For an invertible matrix this decomposition is unique. Using the LDL decomposition Eqn. 2.30 is solved in three steps. Forward substitution is performed in the first step to solve the linear system of equations $\mathtt{L}\mathbf{z} = \mathbf{d}$ for $\mathbf{z}$. Then, $\mathtt{D}\mathbf{y} = \mathbf{z}$ is solved for $\mathbf{y}$ by inverting the diagonal matrix $\mathtt{D}$ which leads to component-wise division with the diagonal entries of $\mathtt{D}$. Finally, $\mathtt{L}^\top\mathbf{x} = \mathbf{y}$ is solved for $\mathbf{x}$ by back substitution.

Furthermore, depending on the structure of $\mathtt{A}$ and hence, also of $\mathtt{C}$ different implementations for solving Eqns. 2.29 and 2.30 can be used. If $\mathtt{A}$ is dense then $\mathtt{C}$ is dense as well. Similarly if $\mathtt{A}$ is sparse then $\mathtt{C}$ is sparse as well. Sparse solvers exploiting the sparse structure of the system matrix are preferable, since they can reduced the computation time significantly.

However, not all optimization and minimization problems that arise are linear and can therefore be solved in the way mentioned above. In the case of solving nonlinear overdetermined systems of equations this leads to nonlinear least squares optimization.

2.2.2 Gauss-Newton Method and Nonlinear Least Squares

The goal in nonlinear least squares optimization, just as in linear least squares optimization, is to solve Eqn. 2.28. However, the cost or error function f cannot be expressed as a convenient matrix. While the solution to a linear system of equations can be solved in one step by inverting the system matrix this is in general not true for nonlinear optimization problems any more. Except for rare cases where a analytic solution exist, iterative solution schemes have to be used in order to compute a parameter vector $\hat{\mathbf{x}}$ that minimizes the error $||\boldsymbol{\varepsilon}|| = \left|\left|f(\hat{\mathbf{x}}) - \tilde{\mathbf{b}}\right|\right|$. Here, we will briefly present the Gauss-Newton method.

The idea behind this method is the following: the function f is approximated linearly at an initial point $\hat{\mathbf{x}}^{(0)}$ such that a linear system of equations arises. Given the solution $\hat{\mathbf{x}}^{(k)}$ a better solution $\hat{\mathbf{x}}^{(k+1)}$ is repeatedly computed until no significantly better solution is found.

Starting with a good initial estimate $\hat{\mathbf{x}}^{(0)}$ we improve it in the following way. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ can be expressed as an infinite sum of values of the function's derivatives evaluated at the expansion point $\mathbf{a} \in \mathbb{R}^n$ by a Taylor series:

$$f(\mathbf{x}) = f_{\mathbf{a}}(\mathbf{x}) = \sum_{|\mathbf{n}|\geqslant 0} \frac{(\mathbf{x}-\mathbf{a})^{\mathbf{n}}}{\mathbf{n}!} D^{\mathbf{n}} f(\mathbf{a}), \tag{2.38}$$

using multi-index notation for $\mathbf{n} = (n_1, \ldots, n_n)^\top \in \mathbb{N}_0^n$:

$$|\mathbf{n}| = \sum_{i=1}^{n} n_i, \quad (\mathbf{x}-\mathbf{a})^{\mathbf{n}} = \prod_{i=1}^{n} (x_i - a_i)^{n_i}, \quad \mathbf{n}! = \prod_{i=1}^{n} n_i!, \quad D^{\mathbf{n}} = \frac{\partial^{|\mathbf{n}|}}{\partial x_1^{n_1} \cdots \partial x_n^{n_n}}. \tag{2.39}$$

Given an estimate $\mathbf{x}$ the function f evaluated at $\mathbf{x} + \boldsymbol{\delta}$ can be linearly approximated according to the Taylor expansion (Eqn. 2.38) around $\mathbf{x}$ by

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{x}+\boldsymbol{\delta}) &= \sum_{|\mathbf{n}|\geqslant 0} \frac{\boldsymbol{\delta}^{\mathbf{n}}}{\mathbf{n}!} D^{\mathbf{n}} f(\mathbf{x}) \\ &= f(\mathbf{x}) + \sum_{i=1}^{n} \delta_i \frac{\partial f(\mathbf{x})}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \delta_j \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} + \ldots \\ &\approx f(\mathbf{x}) + \sum_{i=1}^{n} \delta_i \frac{\partial f(\mathbf{x})}{\partial x_i} \\ &= f(\mathbf{x}) + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \boldsymbol{\delta} \quad = \quad f(\mathbf{x}) + \mathtt{J}(\mathbf{x})\boldsymbol{\delta}, \end{aligned} \tag{2.40}$$

where $\mathtt{J}(\mathbf{x})$ is the Jacobi matrix of f evaluated at $\mathbf{x}$. With $\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}$ and

$\boldsymbol{\varepsilon}^{(n)} = f(\hat{\mathbf{x}}^{(n)}) - \tilde{\mathbf{b}}$ we have

$$\begin{aligned}
\boldsymbol{\varepsilon}^{(n+1)} &= f(\hat{\mathbf{x}}^{(n+1)}) - \tilde{\mathbf{b}} \\
&= f(\hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}) - \tilde{\mathbf{b}} \\
&\overset{\text{Eqn. 2.38}}{=} f_{\hat{\mathbf{x}}^{(n)}}(\hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}) - \tilde{\mathbf{b}} \\
&\overset{\text{Eqn. 2.40}}{\approx} f(\hat{\mathbf{x}}^{(n)}) + \mathtt{J}(\hat{\mathbf{x}}^{(n)})\boldsymbol{\delta}^{(n)} - \tilde{\mathbf{b}} \\
&= \boldsymbol{\varepsilon}^{(n)} + \mathtt{J}(\hat{\mathbf{x}}^{(n)})\boldsymbol{\delta}^{(n)}. \qquad (2.41)
\end{aligned}$$

In order to minimize the error given an initial estimate $\hat{\mathbf{x}}^{(0)}$ we solve the overdetermined linear system of equations

$$\mathtt{J}(\hat{\mathbf{x}}^{(n)})\boldsymbol{\delta}^{(n)} = -\boldsymbol{\varepsilon}^{(n)} \qquad (2.42)$$

for $\boldsymbol{\delta}^{(n)}$, which corresponds to a linear least squares problem. As described in Sec. 2.2.1, Eqn. 2.30 the solution can be obtained by solving the normal equation

$$\mathtt{J}^\top \mathtt{J}\boldsymbol{\delta} = -\mathtt{J}^\top \boldsymbol{\varepsilon} \qquad (2.43)$$

for $\boldsymbol{\delta}$. Finally we obtain the update rule

$$\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}. \qquad (2.44)$$

This iteration scheme using Eqn. 2.43 to compute updates for $\hat{\mathbf{x}}$ in every iteration is called the Gauss-Newton method. Because of the nonlinearity of f local minima can exist. This iterative scheme can converge towards such a local minimum. In order to avoid this from happening the initial estimate for the solution $\hat{\mathbf{x}}^{(0)}$ has to be "close" to the true minimum in order to converge towards the global minimum of f. Usually an approximation is used as an initial estimate that is known to be close to the true minimum. As mentioned above, this scheme is iterated until no significantly better solution is found. This can be achieved by computing $\left|\left|\boldsymbol{\varepsilon}^{(n+1)} - \boldsymbol{\varepsilon}^{(n)}\right|\right|$, the length of the difference of consecutive errors. If the length of this difference vector falls below a specified threshold the iterative scheme terminates. This can be done because f is assumed to be convex at least in the neighborhood of a minimum.

2.2.3 Gradient Descent

Another possibility to minimize Eqn. 2.28 is to compute $\hat{\mathbf{x}}$ using the gradient descent method. The gradient grad f for a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is a directional derivative

$$\text{grad } f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^\top \tag{2.45}$$

and points to the direction of the steepest ascent of the function values of f. Furthermore, the gradient magnitude $|\text{grad } f|$ of the gradient is a measure of how fast those function values increase in the direction of the gradient. The strategy to follow the opposite direction of grad f in order to find the minimum of the function f is called *gradient descent*. Similar to nonlinear least squares optimizations the initial estimate is iteratively updated as in Eqn. 2.44. In the case of gradient descent the update $\boldsymbol{\delta}^{(n)}$ is computed as

$$\lambda \boldsymbol{\delta}^{(n)} = -\text{grad } f(\hat{\mathbf{x}}^{(n)}), \tag{2.46}$$

where $0 < \lambda \leqslant 1$ is the parameter controlling the step size.

In the case of a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ the gradient becomes the Jacobi matrix $\mathtt{J}$ and the update can be computed as

$$\lambda \boldsymbol{\delta}^{(n)} = -\mathtt{J}(\hat{\mathbf{x}}^{(n)})^\top \boldsymbol{\varepsilon}^{(n)}, \tag{2.47}$$

or shortened as

$$(\lambda \mathtt{I}_{n \times n} \boldsymbol{\delta} =)\ \lambda \boldsymbol{\delta} = -\mathtt{J}^\top \boldsymbol{\varepsilon} \tag{2.48}$$

in each iteration. Note the similarity of this equation to Eqn. 2.43, where $\mathtt{J}^\top \mathtt{J}$ is approximated by the identity matrix $\lambda \mathtt{I}_{n \times n}$. Furthermore, for minimizing Eqn. 2.28 in a least squares sense with

$$f(\hat{\mathbf{x}}) = \left|\left| \mathtt{A}\hat{\mathbf{x}} - \tilde{\mathbf{b}} \right|\right|^2, \tag{2.49}$$

we have

$$\text{grad } f(\hat{\mathbf{x}}) = 2\mathtt{A}^\top \left(\mathtt{A}\hat{\mathbf{x}} - \tilde{\mathbf{b}} \right) = 2 \left(\mathtt{A}^\top \mathtt{A}\hat{\mathbf{x}} - \mathtt{A}^\top \tilde{\mathbf{b}} \right). \tag{2.50}$$

From this equation and the necessary condition for a minimum, grad $f(\hat{\mathbf{x}}) = 0$, we obtain the normal equation Eqn. 2.30.

Similar to nonlinear least squares optimization the initial estimate $\hat{\mathbf{x}}^{(0)}$ has to be "close" to the global minimum of f to avoid running into a local minimum. The choice of the step

size influences the required number of iterations of the gradient descent method to converge to the minimum. In some cases the optimal step size λ for each iteration can be derived analytically. In cases where an optimal step size cannot be derived analytically usually a small constant value for λ is used. Iteration terminates if the update does not reduce the cost significantly, so when $\left|\left|\boldsymbol{\varepsilon}^{(n+1)} - \boldsymbol{\varepsilon}^{(n)}\right|\right|$ falls below a specified threshold.

In practice gradient descent is not used very often, because of its slow convergence and running time depending on the value of the parameter λ. Preconditioning can improve the running time of gradient descent. Faster methods for solving linear systems of equations are the conjugate gradient method and the preconditioned conjugate gradient method. Both are related to the gradient descent method.

2.2.4 Levenberg-Marquardt

The Levenberg-Marquardt [97, 109, 115, 63] algorithm combines the methods of Gauss-Newton and gradient descent. It is a iterative method to compute the $\hat{\mathbf{x}}$ that minimizes Eqn. 2.28. Given an initial estimate $\hat{\mathbf{x}}^{(0)}$, better estimates are computed successively by $\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}$ (Eqn. 2.44) using the augmented normal equation

$$\left(\mathtt{J}^\top \mathtt{J} + \lambda \mathtt{I}_{n\times n}\right) \boldsymbol{\delta} = -\mathtt{J}^\top \boldsymbol{\varepsilon} \tag{2.51}$$

to compute the update $\boldsymbol{\delta}^{(n)}$ in each iteration for a specific $0 < \lambda$. The initial value for the parameter $\lambda^{(0)}$ is chosen much smaller than the average of the diagonal elements of $\mathtt{J}^\top \mathtt{J}$. In each iteration Eqn. 2.51 is solved for $\boldsymbol{\delta}^{(n)}$. Then, the error $||\boldsymbol{\varepsilon}^{(n)}|| = ||f(\hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}) - \tilde{\mathbf{b}}||$ is computed. Now, two cases can occur:

1. If $||f(\hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}) - \tilde{\mathbf{b}}|| < ||f(\hat{\mathbf{x}}^{(n)}) - \tilde{\mathbf{b}}||$, then the algorithm has been able to further minimize the error. It proceeds with the next iteration using $\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}$ and a new, smaller $\lambda^{(n+1)} = \lambda_{\text{inc}}^{-1} \cdot \lambda^{(n)}$, where λ_{inc} is the increment factor for λ, typically chosen to be 10.

2. However, if $||f(\hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}) - \tilde{\mathbf{b}}|| \geqslant ||f(\hat{\mathbf{x}}^{(n)}) - \tilde{\mathbf{b}}||$, the algorithm has not been able to minimize the error. In this case Eqn. 2.51 is repeatedly solved for $\boldsymbol{\delta}^{(n)}$ with increasing $\lambda^{(n)} = \lambda_{\text{inc}} \cdot \lambda^{(n)}$, until an update $\boldsymbol{\delta}^{(n)}$ has been found that minimizes the error. The algorithm proceeds with $\hat{\mathbf{x}}^{(n+1)} = \hat{\mathbf{x}}^{(n)} + \boldsymbol{\delta}^{(n)}$ and the new, larger $\lambda^{(n+1)}$ that has been used to find $\boldsymbol{\delta}^{(n)}$.

As with the Gauss-Newton and the gradient descent method the algorithm terminates

whenever no significant reduction of the error is obtained, in example when $\left|\left|\boldsymbol{\varepsilon}^{(n+1)} - \boldsymbol{\varepsilon}^{(n)}\right|\right|$ is below a specified threshold.

This strategy lets the Levenberg-Marquardt algorithm move smoothly between a Gauss-Newton method and the gradient descent method: If the update $\boldsymbol{\delta}^{(n)}$ minimizes the error, $\lambda^{(n)}$ is decreased by the factor λ_{inc}. So, as long as the new update always reduces the error $\lambda \to 0$ and hence, $\mathtt{J}^\top\mathtt{J} + \lambda\mathtt{I}_{n\times n} \to \mathtt{J}^\top\mathtt{J}$. Conversely, if the update does not reduce the error, $\lambda^{(n)}$ is increased by the factor λ_{inc}. So, for $\lambda \to \infty$, we have $\mathtt{J}^\top\mathtt{J} + \lambda\mathtt{I}_{n\times n} \to \lambda\mathtt{I}_{n\times n}$ in the sense that the solution of Eqn. 2.51 with $\mathtt{J}^\top\mathtt{J} + \lambda\mathtt{I}_{n\times n}$ converges to the solution of Eqn. 2.48 for $\lambda \to \infty$.

Thus, as long as the error is reduced the augmented normal equation converges to the normal equation and as long as the error is not reduced the augmented normal equation converges to the equation of the gradient descent method. This combination makes the Levenberg-Marquardt algorithm converge faster toward a minimum, because of the fast convergence of the Gauss-Newton method on the one hand. On the other hand, it guarantees in every step a minimization of the error because of the gradient descent approach. Although the method is slower than a pure Gauss-Newton approach, this makes the method faster then a pure gradient descent approach. Additionally, the method is more robust, because it is better in avoiding local minima as well as less sensitive to the choice of the initial estimate $\hat{\mathbf{x}}^{(0)}$.

Other optimization methods include conjugate gradient (CG) and preconditioned conjugate gradient descent methods (PCG) [72, 40, 150, 140, 15]. Some approaches also use a combination of the Levenberg-Marquardt algorithm with conjugate gradient methods [26, 167, 2].

2.2.5 Random Sample Consensus

The methods for solving Eqn. 2.27 presented so far assumed that the measurements contain small errors. Parameters $\hat{\mathbf{x}} \in \mathbb{R}^n$ are estimated by finding a solution $f(\hat{\mathbf{x}})$ that best approximates the measurements $\tilde{\mathbf{b}} \in \mathbb{R}^m$. In this case all measurements are considered. Concerning the case where some measurements contain large errors this strategy is not advisable any more. A single outlier can render such estimations void.

In this case it is useful to apply the *random sample consensus* algorithm (RANSAC, [47]). The steps of the RANSAC algorithm can be summarized as follows:

1. A minimal sets of measurements is randomly picked. This is a set from which unique

parameters of a model can be computed. From each chosen minimal set of measurements the parameters of the model are computed.

2. From all given measurements, the inlier set of measurements is determined. This is the set of measurements whose distance from the computed model is below a given threshold t_{dist}.

3. If the size of the inlier set is above another threshold t_{size} the model is estimated using all measurements from the inlier set and the algorithm terminates. Otherwise, the steps above are repeated at most $n_{\max}$ times. After $n_{\max}$ minimal sets has been randomly chosen the one with largest inlier set is used to estimate the parameters of the model.

The RANSAC parameter t_{dist} defines which measurements should be considered as inliers and which ones as outliers. The parameter t_{size} defines the minimal size of the inlier set that should be used for model parameter estimation. However, if such a set cannot be found with $n_{\max}$ trials the largest inlier set is used. Since it is not possible to check all possible minimal sets, a maximal number of trials $n_{\max}$ is set. This number should be chosen in a way to ensure with high probability that all measurements in the inlier set are indeed true inliers. For more details about the choice of parameters of the RANSAC algorithm, see also [64]. For other methods dealing with parameter estimation from data sets containing outliers, see for example [130, 32, 135].

2.3 Reconstruction

By acquiring an image the 3D information of a scene is projected onto the 2D image plane. During this projection process the depth information is lost. 3D reconstruction tries to recover this lost information. Although 2D images provide data for each pixel, the lost depth information usually cannot be reconstructed completely. Typically, 3D point clouds are reconstructed and with additional assumptions and constraints dense 3D reconstructions can be obtained. Using 3D point clouds reconstructions can also be generated by fitting surfaces to the point clouds.

Methods for 3D reconstruction can be classified either as active or passive methods. Active methods are used to generate highly accurate, dense 3D reconstructions. They actively manipulate the scene observed with one camera or multiple cameras. This can involve laser triangulation, where a laser line scans the scene and a calibrated camera

observes the 2D projection of the laser line. Structured light approaches use several patterns that are being projected onto the scene and captured by a calibrated camera. Shape-from-shading and photometric stereo methods illuminate the object from different directions in order to reconstruct surface orientations. They are usually used in conjunction with structured light approaches to obtain better reconstructions.

On the other hand, passive reconstruction techniques do not manipulate the scene actively, but only rely on the available scene information. In multi-view stereo reconstruction images of the scene acquired at the same time from several cameras at different positions are used. Using these images of the scene correspondences between different views are established. Based on these correspondences the 3D points can be triangulated and hence their depth and 3D position can be computed. Structure-from-motion is a similar approach to multi-view stereo reconstruction. However, in this approach an image sequence of the scene is captured by a moving camera and used for reconstruction. Because the individual frames of the image sequence are not captured simultaneously only the static content of the scene can be reconstructed in the same way as it is done with multiple view stereo reconstruction. Extensions to structure-from-motion methods exist that enable them to reconstruct moving objects in the image sequence, too.

For an overview of 3D reconstruction methods and is applications is given in [152, 92].

Feature Detection and Matching

Multi-view stereo and structure-from-motion methods rely on *features* that can be easily, reliably and accurately detected and matched within different views. There are different kinds of features, for example lines or points. In the context of feature detection, lines are called edges and points are called corner, as will be described below. Depending on the application, features should not only be detected easily, but also matched easily. While the first task is referred to as *feature detection*, the latter is called *feature matching*. In multiple view stereo reconstruction features are computed for all images, a *feature descriptor* is then computed for all features, and finally, the features are matched based on similarity of their descriptors. On the other hand, in structure-from-motion, first, features are detected for one view and then, similar image patches that contain the detected features are sought in the other views.

Finding 2D-2D correspondences $\mathbf{x}_{i,j} \leftrightarrow \mathbf{x}_{i,j'}$ across different views j and j' for a 3D point $\mathbf{X}_i$ in different views is not possible for every pixel. This is because from different positions of the cameras the corresponding views are different: Parts of the scene may only

be visible in one view because of occlusion. Then, corresponding points for this part of the image do not exist in the other view. If we consider the case of an image sequence captured by a camera, subsequent views are similar. But even in this case correspondences can only be found for some points, because of the *aperture problem*: Point correspondences can be uniquely established for corner points only.

For establishing correspondences between images corner points have to be found. A popular choice to do that is the *Harris corner detector* [61]. It is invariant under rotation and additive illumination changes besides being invariant under translations. Another, more recent, approach for generating correspondences is the *Scale-Invariant Feature Transform* (SIFT, [103, 104]). SIFT is a very popular and widely used feature descriptor, that is invariant under translation, rotation, scaling, additive and multiplicative illumination changes. Other feature detectors are for example: *Speeded Up Robust Features* (SURF, [12, 11]) for corners, the *Canny-Edge Detector* [27] or the *Sobel Operator* [57] for edges.

2.3.1 Multiple View Stereo and Structure-from-Motion

Stereo reconstruction [136] is a passive reconstruction method that uses images captured from two cameras in order to reconstruct the lost depth information. Multiple view stereo reconstruction (MVS, [63, 153, 163, 145, 138, 55, 52, 2]) is an extension of this approach to use more than two images for the reconstruction.

Structure-from-Motion (SfM) is related to multi-view stereo. While in MVS the scene is reconstructed from multiple images, in SfM it is reconstructed from an image sequence of a moving camera.

In both approaches the 3D points $\hat{\mathbf{X}}_i$ from the scene are reconstructed using only feature point correspondences $\tilde{\mathbf{x}}_{i,j} \leftrightarrow \tilde{\mathbf{x}}_{i,j'}$ in different views j, j'. The feature point correspondences are obtained from feature detection, matching, and tracking, or feature descriptors as described above.

Epipolar Geometry and Fundamental Matrix

The underlying geometric constraints on the projections of 3D scene points $\mathbf{X}_i$ to 2D image points $\mathbf{x}_{i,j}$ are called *epipolar geometry* [43, 64]. We will describe the epipolar geometry for the stereo approach, where we have two views.

Given a calibrated camera pair with camera matrices $\mathtt{P}_1, \mathtt{P}_2$, for a 2D point $\mathbf{x}_1$ in the

first view the line of sight $\mathbf{l}_1$ of this point can be computed as

$$\mathbf{l}_1 = \mathtt{P}_1^+ \mathbf{x}_{i,1}, \tag{2.52}$$

where $\mathtt{P}_1^+$ is the pseudo-inverse of $\mathtt{P}_1$: $\mathtt{P}_1\mathtt{P}_1^+ = \mathtt{I}_{3\times3}$. Every point on the line of sight $\mathbf{l}_1$ is projected to $\mathbf{x}_1$ in the first view. This line of sight $\mathbf{l}_1$ can be projected into the second view and it is imaged as a line $\mathbf{l}_2 = \mathtt{P}_2\mathbf{l}_1$ in the image plane, the *epipolar line*. The point $\mathbf{x}_2$ corresponding to $\mathbf{x}_1$ must lie on the epipolar line $\mathbf{l}_2$ and vice versa. Because all line of sights intersect in the camera center $\mathbf{C}$, all epipolar lines intersect in one point $\mathbf{e}$, the *epipole*. The epipoles $\mathbf{e}_1, \mathbf{e}_2$ are the projections of the camera centers $\mathbf{C}_1, \mathbf{C}_2$ onto the image planes of the other views. However, the epipoles do not necessarily have to lie in the image.

The *fundamental matrix* $\mathtt{F}$ [44, 62, 158] is the matrix that relates two views as described above by epipolar geometry. Because the fundamental matrix establishes a relationship between points and lines in the two-dimensional images, no 3D information is needed. Hence, the cameras corresponding to those two views do not have to be calibrated.

Given a measured 2D point $\tilde{\mathbf{x}}_1$ in the first view it allows to compute the epipolar line $\mathbf{l} = \mathtt{F}\tilde{\mathbf{x}}_1$ in the other view (unless the 2D point $\tilde{\mathbf{x}}_1$ coincides with the epipole $\mathbf{e}$). Since the corresponding point $\tilde{\mathbf{x}}_2$ must lie on the epipolar line, we can write:

$$\mathbf{x}'^{\top}\mathtt{F}\tilde{\mathbf{x}} = \mathbf{0}, \tag{2.53}$$

which is called the *epipolar constraint*. The fundamental matrix is defined up to scale and because it maps points in one image to lines in the other image, one eigenvalue is 0, hence $\det(\mathtt{F}) = 0$. Hence, the fundamental matrix has rank 2 and 7 degrees of freedom.

For estimating the fundamental matrix approaches as described in [43, 64] can be used: It is estimated in a least squares sense from established point correspondences and the rank 2 property is enforced in the second step. For estimating a robust fundamental matrix that is not influenced by outliers, the RANSAC algorithm as described in Sec. 2.2.5, [47] can be used. In this case it is advisable to use the *7-point-alogrithm* [71, 148] that allows estimating the fundamental matrix from 7 corresponding point pairs.

Furthermore, the *essential matrix* [101] relates two views exactly in same way as the fundamental matrix, except that the image coordinates are normalized. Image coordinates are normalized, if the calibration matrices of the two cameras are equal to identity matrix: $\mathtt{K}_1 = \mathtt{K}_2 = \mathtt{I}_{3\times3}$.

Projective Reconstruction

Given an estimated fundamental matrix $\hat{\mathtt{F}}$ and a 2D feature point correspondence $\tilde{\mathbf{x}}_1 \leftrightarrow \tilde{\mathbf{x}}_2$ from which $\hat{\mathtt{F}}$ has been estimated, a projective reconstruction of the feature points can be computed in the following way:

1. From the fundamental matrix $\hat{\mathtt{F}}$ the camera matrices for both cameras can be chosen that relate the two views to each other and fulfill the epipolar constraint 2.53 for corresponding points. For $\hat{\mathtt{F}}$ and the epipole $\hat{\mathbf{e}}_2$ in the second image

$$\hat{\mathtt{P}}_1 = \begin{bmatrix} \mathtt{I}_{3\times 3} & \mathbf{0} \end{bmatrix}, \quad \text{and} \quad \mathtt{P}_2 = \begin{bmatrix} [\hat{\mathbf{e}}_2]_\times \hat{\mathtt{F}} & \hat{\mathbf{e}}_2 \end{bmatrix} \tag{2.54}$$

   can be chosen as camera matrices.

2. Compute the depth of the scene points $\hat{\mathbf{X}}_i$ corresponding to its 2D projections $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$. This can be done either by triangulation or by minimizing the reprojection error.

   For triangulation from Eqn. 2.14 we get $\tilde{\mathbf{x}}_1 = \hat{\mathtt{P}}_1\hat{\mathbf{X}}$ and $\tilde{\mathbf{x}}_2 = \hat{\mathtt{P}}_2\hat{\mathbf{X}}$, which can be written as $\tilde{\mathbf{x}}_1 \times \hat{\mathtt{P}}_1\hat{\mathbf{X}} = \mathbf{0}$ and $\tilde{\mathbf{x}}_2 \times \hat{\mathtt{P}}_2\hat{\mathbf{X}} = \mathbf{0}$. This way we obtain 6 constraints in the unknowns of $\hat{\mathbf{X}}$, where 4 constraints are linearly independent. Since $\hat{\mathbf{X}}$ has 3 degrees of freedom, this leads to an overdetermined linear system of equations $\mathtt{A}\hat{\mathbf{X}} = \mathbf{0}$. A solution can be obtained as the singular vector corresponding to the smallest singular value in an SVD (Eqn. 2.33).

   A better approach for estimating the 3D position $\hat{\mathbf{X}}$ is to minimize the reprojection error in both views. This means to minimize the cost function

$$c(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) = d\left(\tilde{\mathbf{x}}_1, \hat{\mathbf{x}}_1\right)^2 + d\left(\tilde{\mathbf{x}}_2, \hat{\mathbf{x}}_2\right)^2, \quad \text{subject to } \hat{\mathbf{x}}_2^\top \hat{\mathtt{F}} \hat{\mathbf{x}}_1 = 0,$$

   with $\hat{\mathbf{x}}_1 = \hat{\mathtt{P}}_1\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}_2 = \hat{\mathtt{P}}_2\hat{\mathbf{X}}$. Here, $d\left(\mathbf{a}, \mathbf{b}\right) = \sqrt{||\mathbf{a} - \mathbf{b}||}$ is the Euclidean distance. $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$ are the measured 2D projections and $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$ are the estimated 2D positions in the image of the estimated 3D point $\hat{\mathbf{X}}$ in the two images. Furthermore, given the estimated fundamental matrix $\hat{\mathtt{F}}$, $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are the estimated 2D positions closest to $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$, respectively, that fulfill the epipolar constraint 2.53. This cost function can be minimized by the Levenberg-Marquardt approach described in Sec. 2.2.4.

   In addition, the camera matrices $\hat{\mathtt{P}}_j$ can be optimized at the same time. This simulta-

neous optimization process is also known as *bundle adjustment* (BA, [160, 102, 167]):

$$\arg\min_{\hat{\mathtt{P}}_j, \hat{\mathbf{X}}_i} \sum_j \sum_i d\left(\tilde{\mathbf{x}}_{i,j}, \hat{\mathtt{P}}_j \hat{\mathbf{X}}_i\right)^2. \tag{2.55}$$

The obtained reconstruction of the 3D points $\hat{\mathbf{X}}_i$ and camera matrices $\hat{\mathtt{P}}_j$ is called projective reconstruction, because it is defined up to a projective transformation given by an invertible homography $\mathtt{H}_{4\times4} \in \mathbb{R}^{4\times4}$. Applying this homography to the camera matrices and 3D points does not change the projections $\hat{\mathbf{x}}_{i,j}$:

$$\hat{\mathbf{x}}_{i,j} = \hat{\mathtt{P}}_j \hat{\mathbf{X}}_i = \left(\hat{\mathtt{P}}_j \mathtt{H}_{4\times4}\right)\left(\mathtt{H}_{4\times4}^{-1} \hat{\mathbf{X}}_i\right) = \hat{\mathtt{P}}'_j \hat{\mathbf{X}}'_i \quad \forall i, j. \tag{2.56}$$

However, given additional information the intrinsic camera parameters can be estimated from several images. This process is called auto-calibration.

Auto-Calibration

Auto-calibration [113, 45, 125, 106] refers to the task of estimating camera parameters without using a calibration object. The intrinsic camera parameters can be estimated given the projective reconstruction of the 3D points $\hat{\mathbf{X}}_i$ and some additional knowledge. Each of the five intrinsic camera parameters focal length f, aspect ratio α, skew s, and principal point offsets p_x, p_y can vary, be fixed, or even be known between subsequent uncalibrated images. Usually both the two principal point offset parameters p_x, p_y vary, are fixed, or are known at the same time. The additional knowledge of a parameter being fixed or known allows to auto-calibrate the cameras. A known parameter provides one constraint for each image in the image sequence. A fixed, but unknown parameter provides one constraint for each image in the image sequence, too, except for the first image.

Usually the camera matrices $\mathtt{P}_j$ form the projective reconstruction will not fulfill these constraints. However, an invertible homography $\mathtt{H}_{4\times4} \in \mathbb{R}^{4\times4}$ as described in Eqn. 2.56 can be computed such that the constraints are fulfilled while the projections of the 3D points $\hat{\mathbf{X}}_i$ do not change.

In order to determine such a homography $\mathtt{H}_{4\times4}$ and perform auto-calibration, at least 8 constraints are needed. This can be seen from the fact, that $\mathtt{H}_{4\times4}$ is defined up to scale and has 15 degrees of freedom, where we have to subtract 7 degrees of freedom for the ambiguity of reconstruction. For the sake of simplicity we can simply assume that the first camera is located in the origin of the world coordinate system, aligned with the coordinate

axes and the scale of the scene is 1. This fixes the orientation, the position, and the scale of the scene in the world coordinate system and hence, leaves 8 degrees of freedom for $\mathtt{H}_{4\times4}$.

Instead of computing the homography $\mathtt{H}_{4\times4}$ for auto-calibration, entities from projective geometry can be used to obtain the calibration matrix. These entities involve the image of the absolute conic, and its dual image, as well as the absolute dual quadric [159, 127, 126].

The image of the absolute conic is the intersection of the absolute conic with the plane at infinity. The absolute conic is an algebraic 2D curve that can be represented by a polynomial of degree 2. The image of the absolute conic is the intersection of the absolute conic with the plane at infinity. The plane at infinity is the 3D analogue to the vanishing line in 2D. The plane at infinity can be thought of as the plane where all possible pairs of parallel lines intersect. Since the image of the absolute conic is located at the plane of infinity, it is independent of the orientation and the position of the camera, as well as the scale. It represents the 5 degrees of freedom that correspond to the intrinsic parameters of the camera.

Metric Reconstruction

Metric reconstruction refers to the reconstruction of the 3D points $\hat{\mathbf{X}}_i$ and camera matrices $\hat{\mathtt{P}}_j$, where the cameras $\hat{\mathtt{P}}_j$ have the form according to Eqn. 2.11, rather than form according to Eqn. 2.14. The cameras are in metric space rather than in projective space. As shown in Eqn. 2.56 a homography $\mathtt{H}_{4\times4} \in \mathbb{R}^{4\times4}$ can be applied to a projective reconstruction without changing the projections $\hat{\mathbf{X}}_i$. A metric reconstruction is obtained from a projective reconstruction by determining and applying the *rectifying homography* $\mathtt{H}_{4\times4} \in \mathbb{R}^{4\times4}$ to a projective reconstruction. The rectifying homography is the homography that makes a metric camera according to Eqn. 2.11 out of a projective camera with $\hat{\mathtt{P}}'_j = \hat{\mathtt{P}}_j\mathtt{H}_{4\times4}$. At the same time the inverse of the rectifying homography is applied to 3D points $\hat{\mathbf{X}}_i$ with $\hat{\mathbf{X}}'_i = \mathtt{H}^{-1}_{4\times4}\hat{\mathbf{X}}_i$ to obtain a metric reconstruction.

By auto-calibration the obtained metric reconstruction is defined up to similarity transformations: Given the 2D measurements of the detected feature points alone, the absolute orientation, absolute position, and absolute scale in the world coordinate system cannot be determined. Without additional knowledge of the scene, the metric reconstruction only determines the relative orientation, relative position, and relative scale of the reconstructed scene.

If, however, the cameras are already calibrated, then a metric reconstruction can be performed directly without having to do a projective reconstruction first, followed by auto-

calibration.

The reconstruction process with detecting corresponding features, computation of the fundamental matrix, projective reconstruction, auto-calibration, and metric reconstruction can be extended to more than two views.

Structure-from-Motion

Structure-from-Motion (SfM [157, 147, 48, 126, 99]) is closely related to MVS: Both approaches aim at reconstructing the imaged scene. While in MVS the scene is imaged from different positions that are usually far apart, in SfM an image sequence from a moving video camera is captured.

In addition the intrinsic and extrinsic camera parameters are estimated as well. If the extrinsic parameters are known beforehand [121], only a camera pose estimation has to be performed (see Sec. 2.1.5).

As described above, the quality of the results depends on the ability to detect features and to identify corresponding features between views. In this respect, finding point correspondences in structure-from-motion is easier than with MVS, since individual images of the image sequence are captured from positions that are very close to each other. This is because of the high frame rate of the video camera used to capture the sequence. Consecutive images of the sequence are similar, because the points of view change slightly. Hence, instead on relying on feature descriptors only, detected features in one image can be much easier tracked over the image sequence. In fact, feature detection and feature descriptors are only used to generate new feature points that are tracked in the sequence [105, 156]. New feature points have to be detected in the first image of the sequence and whenever tracked features cannot be found in the next image due to occlusion or change in point of view. The Harris corner detector [61] can be used for detecting features.

The 3D reconstruction is performed in a similar way as with MVS. However, there are a few exceptions. First, because of the high frame rate of the camera the position of the camera in subsequent images does not change much. In this case the fundamental matrix cannot be estimated accurately. It is also possible, that the camera does not move at all and hence, the fundamental matrix cannot be estimated at all. In this case the fundamental matrix simplifies to a homography $\mathtt{H}_{4\times4}$, that represents a rotation of the camera in 3D. Hence for estimating the positions of the 3D points $\hat{\mathbf{X}}_i$ *key frames* have to be chosen, such that a fundamental matrix can be estimated accurately and reliably. The key frames can be chosen according to the *geometric robust information criterion* (GRIC, []), such that the

position of the camera significantly changes between subsequent key frames. Second, the 3D points $\hat{\mathbf{X}}_i$ and camera matrices $\hat{\mathtt{P}}_j$ are estimated for two views by bundle adjustment (BA, see Eqn. 2.55). Then, subsequent key frames are added one by one, where a bundle adjustment is performed for every newly added key frame. Last, all images between the chosen key frames are used to improve the estimate of $\hat{\mathbf{X}}_i$ and $\hat{\mathtt{P}}_j$. This way the camera parameters are estimated as well. Different variants exist in order to improve and speed up the results, for example hierarchical BA, incremental BA, or sliding windows BA.

SfM is related to the problem of *simultaneous localization and mapping* (SLAM, [143, 144, 39, 5]): Generate a map of the surroundings and localize a moving robot within those surroundings using the data of the robot's camera.

Multi-View Stereo and Structure-from-Motion approach only reconstruct features points that can be detected, matched, and tracked. This results a in sparse scene structure, where the reconstructed points correspond to feature points. Given images of an object from multiple calibrated cameras the 3D shape of the object in the form of a dense 3D point cloud can be reconstructed with Shape-from-Silhouettes approaches.

2.3.2 Shape-from-Silhouettes

Shape-from-Silhouettes is a 3D reconstruction approach for objects based on silhouette information [10, 120, 21, 142, 42]. In this approach information from the object's silhouettes in multiple images are used for reconstructing a dense point cloud.

Given an image I of the object acquired by a calibrated camera, the silhouette of the object in the image has to be extracted first. This can be done by acquiring an additional image I_B of the background without object. Subtracting the background image from the image with object leads to an image of the foreground. Then, the *silhouette* in the image is the boundary between the object and the background.

Each point of the silhouette can be back-projected to a ray in 3D space. For each camera the set of all back-projected rays from a silhouette generate a *generalized cone*, which is a volume in 3D space. The generalized cones obtained from each camera can be intersected to generate the 3D volume in which the acquired object must lie.

Given a sufficient amount of views that are distributed around the object one obtains the *visual hull* [94, 124, 149, 112]. This visual hull is a 3D volume that approximates the shape of the acquired object.

In practice the generalized cones can be represented as polyhedra [10, 110]. Computing polyhedron-polyhedron-intersection however, is expensive. Hence, in practice the volume

of the object is discretized into a 3D grid *voxels* (volume elements, [151]), that are the 3D analogue of pixels. The visual hull is then obtained in the following way. Each voxel is projected into all camera views. A voxel that lies within all silhouettes of all views is kept, otherwise the voxel is removed from the grid. In the end the 3D grid only contains voxels that lie within all silhouettes and hence, the grid approximates the shape of the object. Instead of discretizing the 3D volume, the projection of the visual hull in the images can be computed [111].

The quality of the reconstructed shape depends on the size of the voxels. The smaller the voxels the more accurate the shape is. On the other hand, smaller voxels increases the number of voxels that have to be checked, which in turn increases the computation time significantly.

2.3.3 Space Carving

In general, concave regions cannot be obtained in the way described above. However, in the case of *Lambertian reflectance* (see Sec. 2.3.6 and Eqn. 2.70), photometric information can be used to refine the initially obtained visual hull from Shape-from-Silhouette approaches.

Given a visual hull obtained from a Shape-from-Silhouettes approach of an object with Lambertian reflectance, a *space carving* [91, 51] method can be used. Space carving works as follows.

Voxels of the initial 3D grid are checked for *photo-consistency*. The voxel is projected into all views. In the views, where the voxel is visible and hence, not occluded by other voxels of the grid, the color information is compared. If the color in all those views is inconsistent, the voxel is removed from the grid, since the color information must have come from different voxels. This is because the Lambertian reflectance property ensures that the color of a point on the object's surface is independent of the camera position. These steps are repeated for all voxels until only photo-consistent voxels remain.

In this way, the shape of an acquired object is obtained that is closer to the true shape, since not only the information from silhouettes, but also color information is used. However, in the presence of image noise photo-consistent voxels can be classified as photo-inconsistent and vice versa, which limits the ability of space carving methods to accurately reconstructed the true 3D shape.

Other methods for obtaining the true 3D shape of an object given an initial visual hull use the discrete optimization method of (volumetric) graph cuts [23, 88, 89].

In contrast to space carving, methods based on *voxel coloring* [139, 41, 35, 163] do not

remove voxels in which the captured object cannot be, but add photo-consistent voxels across views to obtain a 3D model.

All presented approaches so far used either images from multiple cameras or from an image sequence acquired by a moving camera. In the next section we will describe how the shape of a 3D object can be reconstructed from one view given additional information about scene.

2.3.4 Shape-from-Shading

Shape-from-Shading [74, 78, 50, 150, 173, 38] refers to the task of reconstructing the 3D shape of an object from one view given the position $\boldsymbol{\omega}_i$ of the single light source in the scene. We assume, that the acquired object has Lambertian reflectance (see Sec. 2.3.6 and Eqn. 2.70). However, this does not have to be case in general. Hence, also shape-from-shading methods with non Lambertian reflectance have been developed [6, 96, 3].

In the case of Lambertian reflectance the image irradiance $I(\mathbf{x})$ is proportional to $\mathbf{n} \cdot \boldsymbol{\omega}$ (see also Eqn. 2.71):

$$I(\mathbf{x}) \sim \mathbf{n} \cdot \boldsymbol{\omega} = \cos\theta, \tag{2.57}$$

where θ is the angle between $\mathbf{n}$ and $\boldsymbol{\omega}$. This equation provides one constraint for the two unknowns of the unit surface normal $\mathbf{n}$. Hence, a unique solution cannot be computed without any other assumptions.

For the sake of simplicity, we assume that the object is imaged in the camera coordinate system, such that the camera center $\mathbf{C}$ is located in the origin and the optical axis coincides with the z-axis (see Sec. 2.1.3). Furthermore, we assume the camera to be an orthographic camera. This is a camera that is so far away from the object that the line of sights are equal for all pixels. In this case all possible directions of the normal $\mathbf{n}(\mathbf{x}) = (n_1(\mathbf{x}), n_2(\mathbf{x}), n_3(\mathbf{x}))^\top$ at $\mathbf{x}$ can be represented by points $\mathbf{n}'(\mathbf{x}) = (n'_1(\mathbf{x}), n'_2(\mathbf{x}))^\top$ in a 2D plane with stereographic projection $\mathbf{n}(\mathbf{x}) \mapsto \mathbf{n}'(\mathbf{x})$ given by

$$n'_1 = \frac{2 \cdot n_1}{n_3 - 1}, \quad n'_2 = \frac{2 \cdot n_2}{n_3 - 1} \tag{2.58}$$

and the back-projection $\mathbf{n}'(\mathbf{x}) \mapsto \mathbf{n}(\mathbf{x})$ is given by

$$n_1 = \frac{4 \cdot n'_1}{n'^2_1 + n'^2_2 + 4}, \quad n_2 = \frac{4 \cdot n'_2}{n'^2_1 + n'^2_2 + 4}, \quad n_3 = \frac{n'^2_1 + n'^2_2 - 4}{n'^2_1 + n'^2_2 + 4}, \tag{2.59}$$

Assuming that the reconstructed surface of the object is smooth the shape is reconstructed by minimizing the cost function

$$c(\mathbf{n}'(\mathbf{x})) = \int_\Omega F(n'_1, n'_2, n'_{1,x}, n'_{1,y}, n'_{2,x}, n'_{2,y})\,\mathrm{d}\mathbf{x}, \tag{2.60}$$

with

$$\begin{aligned} F(n'_1, n'_2, n'_{1,x}, n'_{1,y}, n'_{2,x}, n'_{2,y}) &= \left(I(\mathbf{x}) - \mathbf{n}'(\mathbf{x}) \cdot \boldsymbol{\omega}\right)^2 \\ &\quad + \alpha \cdot \left(||\text{grad } n'_1(\mathbf{x})||^2 + ||\text{grad } n'_2(\mathbf{x})||^2\right) \end{aligned} \tag{2.61}$$

and partial derivatives

$$n'_{1,x} = \frac{\partial n'_1}{\partial x}, \quad n'_{1,y} = \frac{\partial n'_1}{\partial y}, \quad n'_{2,x} = \frac{\partial n'_2}{\partial x}, \quad n'_{2,y} = \frac{\partial n'_2}{\partial y}. \tag{2.62}$$

In 2.61, the first term $(I(\mathbf{x}) - \mathbf{n}'(\mathbf{x}) \cdot \boldsymbol{\omega})^2$ is the data term representing the difference between the given image I and the reconstructed surface normals $\mathbf{n}'(\mathbf{x})$. The second term $(||\text{grad } n'_1(\mathbf{x})||^2 + ||\text{grad } n'_2(\mathbf{x})||^2)$ is the smoothness term, which penalizes deviations from a smooth solution of $\mathbf{n}'(\mathbf{x})$. With the parameter α the smoothness of the solution $\mathbf{n}'(\mathbf{x})$ can be controlled. Furthermore, we have the boundary condition, that the normal $\mathbf{n}'(\mathbf{x})$ at point $\mathbf{x}$ that belongs to the silhouette of the object is perpendicular to the line of sight $\mathbf{l}(\mathbf{x})$ and perpendicular to the tangent $\mathbf{t}(\mathbf{x})$ at that point: $\mathbf{n}'(\mathbf{x}) = \mathbf{l}(\mathbf{x}) \times \mathbf{t}(\mathbf{x})$.

Eqn. 2.60 can be minimized using variational calculus. Then, the solution is obtained from the Euler-Lagrange equations

$$F_{n'_1} - \frac{\partial F_{n'_{1,x}}}{\partial x} - \frac{\partial F_{n'_{1,y}}}{\partial y} = 0, \tag{2.63}$$

$$F_{n'_2} - \frac{\partial F_{n'_{2,x}}}{\partial x} - \frac{\partial F_{n'_{2,y}}}{\partial y} = 0 \tag{2.64}$$

corresponding to the cost function 2.60 with the boundary condition $\mathbf{n}'(\mathbf{x}) = \mathbf{l}(\mathbf{x}) \times \mathbf{t}(\mathbf{x})$. Here, $F_\xi = \frac{\partial F}{\partial \xi}$ denote the partial derivative of F with respect to the variable ξ. These Euler-Lagrange equations are coupled partial differential equations. The solution is obtained by solving the linear system of equations in the unknowns $n'_1(\mathbf{x})$ and $n'_2(\mathbf{x})$ in all points $\mathbf{x}$ arising from discretization of Eqns. 2.63 and 2.64 with boundary condition.

2.3.5 Structured Light

In contrast to Shape-from-Shading approaches that use a omnidirectional light source and a smoothness assumption on the reconstructed surface, Structured Light scanning (SL, [128, 16, 86, 137]) approaches use directional light sources. A common set-up uses a pair consisting of a calibrated video projector and a calibrated camera. The usage of a projector makes SL an active reconstruction technique. Several specially encoded images or patterns are projected into the scene and onto the object by the projector and are being captured by the camera. From these images the objects surface can be dense reconstructed without any assumptions being made.

A video projector can be taught of as an inverse camera. For a camera, all 3D points $\mathbf{X}$ that lie on a line of sight are projected to one 2D point $\mathbf{x}$ in the image plane according to Eqn. 2.14: $\mathbf{x} = \mathtt{P}\mathbf{X}$. In contrast, a projector projects a 2D point $\mathbf{x}$ from the image plane to the line $\mathbf{l} = \mathtt{P}^{+}\mathbf{x}$ as described in Eqn. 2.52:

$$\mathbf{X}_d = \mathbf{C} + d \cdot \mathbf{l} = \mathbf{C} + d \cdot \mathtt{P}^{+}\mathbf{x} \quad \text{for } \lambda \in \mathbb{R}^{+}. \tag{2.65}$$

In this case $\mathbf{X}_d$ is the point in 3D space with depth d, given that $\mathbf{l}$ is a unit vector. The projection of the 3D point $\mathbf{X}_d$ that is captured in the image by the camera is the intersection of the line of sight $\mathbf{l}$ corresponding to $\mathbf{x}$ with the object's surface with minimal depth. Other intersections with larger depth cannot be imaged because of occlusion.

In order to reliable identifying corresponding points between the projected images and the captured images, a sequence of patterns is employed [134, 133]. Simple approaches use *binary* patterns [54], *Gray code* patterns [19] or color-coded stripe patterns [28]. Binary and Gray code patterns consists of repeated, regular, black and white stripes with varying width between images. White stripes encode a 1 and black stripes encode a 0 of a binary number. Considering all projected patterns together, a single stripe with the smallest depth is uniquely determined by the patterns because of its binary code. Furthermore, the Gray code is a binary code that differs only in one bit between subsequent encoded values. In order to use the maximal resolution of the projector the stripes with smallest width are one pixel wide. If the projector and camera are arranged horizontally, then those stripes are usually arranged vertically and vice versa.

A line that is projected into the scene corresponds to a plane in the scene. Using binary or Gray code, each line that is projected onto the object is uniquely determine. Hence, each plane in 3D is uniquely determined. Considering all captured images, the

encoding of an imaged pixel $\mathbf{x}$ in the calibrated camera uniquely determines the plane it is lying in. Then, the true 3D position $\mathbf{X}_d$ of the point $\mathbf{x}$ can be obtained by computing a ray-plane-intersection, which leads to the depth d.

Instead of projecting time varying patterns, correspondences between projector and camera pixels can also be established by color [37], or color-coded stripes [22, 172].

In this set-up no smoothness constraints or boundary conditions are needed. However, there are two limitations. Because of occlusions, only parts of the object that are visible from both, the projector and the camera, can be reconstructed. This leads to holes in the reconstructed surface. This occlusions can be minimized be reducing the baseline distance, the distance between the center of the projector and the center of the camera. But then, the accuracy of the computed depth is reduced. Second, the resolution of the reconstruction depends on the resolution of the video projector and the one of the camera. Since projectors usually have a much lower resolution than cameras, the resolution of the reconstructed object is not very high.

In order to generate reconstructions with a high amount of details Structured Light is used in combination with a technique called Photometric Stereo, that we introduce now.

2.3.6 Photometric Stereo

Photometric Stereo (PS, [166, 141, 75, 165, 68]) is an active method for reconstructing high-accuracy details of surfaces. In contrast to Shape-from-Shading methods that has to use smoothness constraints and boundary conditions to obtain a unique solution to Eqn. 2.57 for every pixel, in photometric stereo several light sources are used to solve for the surface normal $\mathbf{n}$. Again, we consider the case of Lambertian reflectance of the object. For approaches dealing with non-Lambertian reflectance, see [77, 116, 154, 70].

Calibrated Photometric Stereo

We assume now, that the light source positions are known with respect to the camera position in the camera coordinate system. This set-up is called *calibrated photometric stereo*. Here, it is assumed that the direction to the light sources and the direction to the camera is the same for all points on the illuminated can captured object. This corresponds to an orthographic camera as described in Sec. 2.1.3.

For a unique solution, three point light sources with known directions given by unit

vectors $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_3$ are used. Then, the following linear system of equations is obtained:

$$\begin{aligned} \rho_d \cdot \boldsymbol{\omega}_1^\top \cdot \mathbf{n}_i &= I_{i,1}(\mathbf{x}), \\ \rho_d \cdot \boldsymbol{\omega}_2^\top \cdot \mathbf{n}_i &= I_{i,2}(\mathbf{x}), \\ \rho_d \cdot \boldsymbol{\omega}_3^\top \cdot \mathbf{n}_i &= I_{i,3}(\mathbf{x}), \end{aligned} \tag{2.66}$$

where ρ_d is the albedo of the surface. In total there are three degrees of freedom: The normal $\mathbf{n}_i$ has two degrees of freedom and the surface albedo ρ_d one additional degree of freedom. These three constraints can be conveniently written in vector-matrix notation as

$$\Omega^\top \cdot \mathbf{N}_i = \mathbf{I}_i, \tag{2.67}$$

with $\mathbf{N}_i = \rho_d \cdot \mathbf{n}_i$. This linear system of equations can be solved for $\mathbf{N}_i \in \mathbb{R}^3$ with a unique solution for each pixel i, if $\mathbf{I}_i \in \mathbb{R}^3$, $\Omega \in \mathbb{R}^{3\times 3}$ and if Ω is invertible. The matrix Ω is invertible if the directions to the three light sources are not collinear. Then we have three linear independent constraints, Ω is invertible with $\Omega \cdot \Omega^\top = \mathtt{I}$. The solution $\mathbf{N}_i$ can be computed by

$$\mathbf{N}_i = \Omega \cdot \mathbf{I}_i. \tag{2.68}$$

On the other hand, if n light sources are used with $n > 3$, we have Eqn. 2.67 with $\mathbf{I}_i \in \mathbb{R}^n$, $\Omega \in \mathbb{R}^{3\times n}$. The linear system of equations is overdetermined. A least squares solution can be computed by SVD as described above (see Eqn. 2.33).

Lambertian Reflectance and Rendering Equation

The luminance $L_o(\mathbf{x}, \boldsymbol{\omega}_o)$ (or observed radiance) of a surface point $\mathbf{x}$ with a given surface normal that is observed or imaged by a camera from viewing direction $\boldsymbol{\omega}_o$ depends on:

1. the amount of emitted light $L_e(\mathbf{x}, \boldsymbol{\omega}_o)$ of that surface point $\mathbf{x}$ in the viewing direction $\boldsymbol{\omega}_o$,

2. the reflection property of the surface at the considered point $\mathbf{x}$, expressed by the *bidirectional reflectance distribution function* (BRDF, [118, 119]) $R(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \mathbf{n})$ which specifies the amount of reflected light in direction $\boldsymbol{\omega}_o$ given the amount of incoming light from direction $\boldsymbol{\omega}_i$,

3. the amount and distribution of incoming light $L_i(\mathbf{x}, \boldsymbol{\omega}_i)$ from all possible directions $\boldsymbol{\omega}_i$ of the unit hemisphere of the considered surface point, and

4. the angle $\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i$ between the orientation $\mathbf{n}(\mathbf{x}) = (n_x(\mathbf{x}), n_y(\mathbf{x}), n_z(\mathbf{x}))^\top$ of the surface normal at point $\mathbf{x}$ and the direction of incoming light $\boldsymbol{\omega}_i$.

The non-negativity of the angle $\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i$ follows from the fact that light that is illuminating the surface from behind is blocked by the object itself and is not contributing to the amount of incoming light. This fact can be easily expressed as $\max(\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i, 0)$.

The mentioned dependencies are expressed in the *rendering equation* [85, 79]:

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o) = L_e(\mathbf{x}, \boldsymbol{\omega}_o) + \int_\Omega R(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \mathbf{n}) \cdot L_i(\mathbf{x}, \boldsymbol{\omega}_i) \cdot \max(\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i, 0) \, \mathrm{d}\boldsymbol{\omega}_i. \tag{2.69}$$

Note that in this equation for the sake of simplicity we do not consider the wavelength spectrum of the incoming and outgoing light, transparent and translucent medium, or subsurface scattering. This implies that while reflections can be expressed by an appropriate BRDF, refraction, transparency, and translucency cannot. Here transparency is the effect of light passing through a medium and being possibly refracted following Snell's law, but not being scattered. While translucency is the effect of light passing through medium being scattered and not following Snell's law. Subsurface scattering on the other hand, is the effect of light entering the material, being scattered once or multiple times beneath the surface and exiting the material close to the point where it has entered the material.

Extensions and generalizations of the BRDF such as the *bidirectional scattering surface reflectance distribution function* (BSSRDF, [119, 83]) for handling subsurface scattering, or the *bidirectional transmittance distribution function* (BTDF, [7]) to handle transparency and translucency, exist to deal with these situations.

This work focuses on materials with Lambertian reflectance [93]. For an object made of a material with Lambertian reflectance property the perceived brightness does not depend on the viewing direction:

$$R(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \mathbf{n}) = \rho_d, \tag{2.70}$$

where ρ_d is the albedo of the surface, specifying the ratio between the reflected and incoming light.

Given an object that does not emit light, that is made of a material with Lambertian reflectance property, and fixing $\mathbf{x}$ the rendering equation 2.69 simplifies to:

$$L_o(\boldsymbol{\omega}_o) = \int_\Omega \rho_d \cdot L_i(\boldsymbol{\omega}_i) \cdot \max(\mathbf{n} \cdot \boldsymbol{\omega}_i, 0) \, \mathrm{d}\boldsymbol{\omega}_i = \rho_d \cdot \int_\Omega L_i(\boldsymbol{\omega}_i) \cdot \max(\mathbf{n} \cdot \boldsymbol{\omega}_i, 0) \, \mathrm{d}\boldsymbol{\omega}_i. \tag{2.71}$$

So, in case of Lambertian reflectance the luminance $L_o(\boldsymbol{\omega}_o)$ is independent of $\boldsymbol{\omega}_o$. Hence,

a point on an object with Lambertian reflectance is imaged with the same luminance independent of the camera position.

Controlled Illumination

If, in addition, the position of the light sources can be chosen the computation of the normal can be further simplified. According to Eqn. 2.71 for an object with Lambertian reflectance property and a known illumination setup, the image irradiance does not depend on the camera position.

We can express the integration over the unit hemisphere Ω in this equation by integrating over all possible directions of the hemisphere. These direction are all the directions of the upper hemisphere Ω^+ in a local coordinate system where the z-axis coincides with the surface normal $\mathbf{n}$ and the x- and y-axis are oriented perpendicular to each and to the z-axis, but otherwise arbitrarily:

$$L_o(\boldsymbol{\omega}_o) = \int_{\Omega} \rho_d \cdot L_i(\boldsymbol{\omega}_i) \cdot \max(\mathbf{n}(\mathbf{x}) \cdot \boldsymbol{\omega}_i, 0)\, \mathrm{d}\boldsymbol{\omega}_i = \int_{\Omega^+} \rho_d \cdot L_i(\mathtt{T}^\top \boldsymbol{\omega}_i') \cdot \mathbf{n} \cdot \boldsymbol{\omega}_i'\, \mathrm{d}\boldsymbol{\omega}_i', \tag{2.72}$$

where

$$\mathtt{T} \cdot \boldsymbol{\omega}_i = \begin{pmatrix} s_x & s_y & s_z \\ t_x & t_y & t_z \\ n_x & n_y & n_z \end{pmatrix} \cdot \begin{pmatrix} \omega_s \\ \omega_t \\ \omega_n \end{pmatrix} = \begin{pmatrix} \omega_s' \\ \omega_t' \\ \omega_n' \end{pmatrix} = \boldsymbol{\omega}_i', \tag{2.73}$$

and $\mathtt{T}^\top \cdot \boldsymbol{\omega}_i' = \boldsymbol{\omega}_i$, which follows from $\mathtt{T} \cdot \mathtt{T}^\top = \mathtt{I}$.

In an environment where the illumination can be controlled the incoming light $L_i(\boldsymbol{\omega}_i')$ from every direction $\boldsymbol{\omega}_i'$ at a considered surface point is known. Assuming that the light is coming from the positive x-direction and that the light is parallel we have $L_i(\mathtt{T}^\top \boldsymbol{\omega}_i') = \omega_s' \cdot s_x + \omega_t' \cdot t_x + \omega_n' \cdot n_x$, and for the luminance $L_o^x(\boldsymbol{\omega}_o)$ of the observed point under illumination coming from the x-direction we have

$$\begin{aligned} L_o^x(\boldsymbol{\omega}_o) &= \int_{\Omega^+} \rho_d \cdot L_i(\mathtt{T}^\top \boldsymbol{\omega}') \cdot (\mathbf{n} \cdot \boldsymbol{\omega}')\, \mathrm{d}\boldsymbol{\omega}' \\ &= \int_{\Omega^+} \rho_d \cdot \left(\omega_s' \cdot s_x + \omega_t' \cdot t_x + \omega_n' \cdot n_x\right) \cdot \mathbf{z} \cdot \boldsymbol{\omega}'\, \mathrm{d}\boldsymbol{\omega}' \\ &= \int_{\Omega^+} \rho_d \cdot \omega_n' \cdot n_x \cdot \omega_n'\, \mathrm{d}\boldsymbol{\omega}' \\ &= \rho_d \cdot n_x \cdot \int_{\Omega^+} {\omega_n'}^2\, \mathrm{d}\boldsymbol{\omega}' \end{aligned} \tag{2.74}$$

Furthermore, to integrate ω_n' over all directions $\boldsymbol{\omega}'$ of the hemisphere Ω^+ we use spher-

ical coordinates instead of Cartesian coordinates. From the transformation of Cartesian coordinates $(x, y, z)^\top$ to spherical coordinates $(r, \theta, \varphi)^\top$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} r \cdot \sin\theta \cdot \cos\varphi \\ r \cdot \sin\theta \cdot \sin\varphi \\ r \cdot \cos\theta \end{pmatrix} \tag{2.75}$$

the Jacobi matrix of the transformation is given by

$$\mathtt{J} = \frac{\partial(x, y, z)^\top}{\partial(r, \theta, \varphi)^\top} = \begin{pmatrix} \sin\theta \cdot \cos\varphi & r \cdot \cos\theta \cdot \cos\varphi & -r \cdot \sin\theta \cdot \sin\varphi \\ \sin\theta \cdot \sin\varphi & r \cdot \cos\theta \cdot \sin\varphi & r \cdot \sin\theta \cdot \cos\varphi \\ \cos\theta & -r \cdot \sin\theta & 0 \end{pmatrix} \tag{2.76}$$

with determinant $\det \mathtt{J} = r^2 \cdot \sin\theta$. This yields $\mathrm{d}V = r^2 \cdot \sin\theta \,\mathrm{d}r \,\mathrm{d}\theta \,\mathrm{d}\varphi$ as volume element and $\mathrm{d}A = \frac{\mathrm{d}V}{\mathrm{d}r} = r^2 \cdot \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\varphi$ as surface element for integration. With $r = 1$ we obtain

$$\mathrm{d}A = \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\varphi \tag{2.77}$$

as surface element for the integration over the unit hemisphere and

$$\omega_n' = \cos\theta. \tag{2.78}$$

Hence,

$$\begin{aligned} L_o^x(\boldsymbol{\omega}_o) &= \rho_d \cdot n_x \cdot \int_{\Omega^+} {\omega_n'}^2 \,\mathrm{d}\boldsymbol{\omega'} \\ &\overset{\text{Eqn. 2.77}}{\underset{\text{Eqn. 2.78}}{=}} \rho_d \cdot n_x \cdot \int_0^{\frac{\pi}{2}} \int_0^{2\cdot\pi} \cos^2\theta \cdot \sin\theta \,\mathrm{d}\varphi \,\mathrm{d}\theta \\ &= 2 \cdot \pi \cdot \rho_d \cdot n_x \cdot \int_0^{\frac{\pi}{2}} \cos^2\theta \cdot \sin\theta \,\mathrm{d}\theta \\ &= 2 \cdot \pi \cdot \rho_d \cdot n_x \cdot \left[-\frac{1}{3} \cdot \cos^3\theta\right]_0^{\frac{\pi}{2}} \\ &= 2 \cdot \pi \cdot \rho_d \cdot n_x \cdot \left(-\frac{1}{3} \cdot \cos^3\left(\frac{\pi}{2}\right) + \frac{1}{3} \cdot \cos^3(0)\right) \\ &= \left(\frac{2}{3} \cdot \pi \cdot \rho_d\right) \cdot n_x, \end{aligned} \tag{2.79}$$

and the luminance of an observed surface point $\mathbf{x}$ with surface normal $\mathbf{n} = (n_x, n_y, n_z)^\top$ under illumination coming from the x-direction is proportional to the x-component of the

surface normal $\mathbf{n}$: $L_o^x(\boldsymbol{\omega}_o) \sim n_x$. With similar argument $L_o^y(\boldsymbol{\omega}_o) \sim n_y$ and $L_o^z(\boldsymbol{\omega}_o) \sim n_z$ hold as well [107].

Uncalibrated Photometric Stereo

Uncalibrated photometric stereo refers to the task of computing the surface normals of a captured object, if the directions to the light sources are not known [8, 9, 46]. In this case the positions of the light sources are computed as well. This means to solve Eqn. 2.67 for $\mathbf{N}_i$ with unknown light directions Ω.

This can be done by considering the information from all pixels and computing the normals for all pixels at the same time. Stacking up those equations for n light sources and m pixels we obtain a linear system of equations

$$\Omega^\top \cdot \mathbf{N} = \mathbf{I}, \tag{2.80}$$

with $\mathbf{N} = \boldsymbol{\rho}_d \cdot \mathbf{n}$ consisting of the stacked vectors $\mathbf{n}_i$ with stacked albedo $\rho_{i,d}$ for each pixel i. Here, $\mathbf{N} \in \mathbb{R}^{3 \cdot m}$, $\mathbf{I} \in \mathbb{R}^{n \cdot m}$ and $\Omega \in \mathbb{R}^{3 \cdot m \times n \cdot m}$. This linear system of equations is overdetermined if $n > 3$. A least squares solution can be computed with SVD (Eqn. 2.33).

However, in the case of uncalibrated photometric stereo the normals and light directions can be determined up to an invertible 3×3 matrix $\mathtt{G}$ for each pixel i. This is called the *generalized bas-relief ambiguity* (GBA, [87, 13]). Considering the scenario with one light source and one normal as given in Eqn. 2.67 a set of solutions exists:

$$\Omega^\top \cdot \mathtt{G}^{-1} \cdot \mathtt{G} \cdot \mathbf{N}_i = \mathbf{I}_i, \tag{2.81}$$

with $\mathtt{G} \in \mathbb{R}^{3 \times 3}$. If we fix a coordinate system where x-, and y-axis coincide with the x-, and y-axis of the image, then the z-axis is parallel to the viewing direction of the camera. In this case the matrix $\mathtt{G}$ can be written as

$$\mathtt{G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & c \end{bmatrix}, \tag{2.82}$$

where a, b, and $c \in \mathbb{R}$. In order to be able to compute the true surface normals and light directions additional assumptions have to be used. This includes assumptions on the surface geometry, the surface albedo, or the strength of the light sources.

Chapter 3

Camera Calibration

This chapter describes how we calibrate multiple cameras. Multiple cameras have to be calibrated geometrically in order to obtain their extrinsic camera parameters, which describe the relative positions and orientations to each other and their intrinsic camera parameters, which describe the process of projection of the scene onto the image plane. These parameters allow to compute correspondences between pixels in the image and line of sights in 3D space. The cameras have to be calibrated photometrically in order to reproduce the same color in the same way. This entails the camera sensor's response to different amounts of incoming light, as well as the relative response of the sensor to the three colors red, green, and blue.

Applications include the calibration of cameras in various 3D reconstruction set-ups, for example in multiple view stereo reconstruction, shape-from-silhouettes approaches, structured-light approaches, but also in the case of calibrating cameras in a motion capture studio, for robot navigation, or for augmented reality applications. We also use camera calibration for 3D reconstruction in the set-up described in the Chapter 4. As described in Sec. 2.3.1, the cameras in multi-view reconstruction can be also auto-calibrated from the acquired data using some assumptions. However, better calibration results can be obtained by performing an explicit camera calibration, if possible.

First, we present an approach for the geometric calibration of the cameras that is an extension of the approach presented in [59]. Then, we show how we perform a photometric calibration of them.

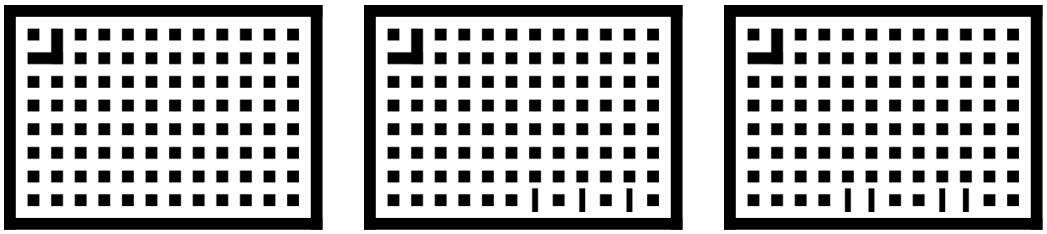

Figure 3.1: Examples of calibration patterns used in our approach. Each pattern consists of a grid with 8 rows and 12 columns. A marker in one corner determines the orientation of the pattern. A pattern can be identified by the binary number encoded in the bottom row. From **left** to **right**: Patterns with identifiers $0, 42 = 101010_{(2)}$, and $204 = 11001100_{(2)}$, respectively.

3.1 Geometric Calibration

In this section we describe our approach to estimate the intrinsic and extrinsic parameters of multiple cameras using multiple coded patterns. The cameras can be static or moving. While estimating camera parameters we automatically obtain parameters for all visible patterns in the image. The parameters of a pattern include the position of the pattern and its orientation in 3D space. They are similar to the extrinsic camera parameters.

3.1.1 Overview

In the approaches of Tsai [161] and Zhang [174] the corners of square regions on the calibration patterns are used as points. Finding those corner points becomes less accurate with smaller size of the pattern in the image. Hence, we use the centers of the regions as initial 2D points as they are easier to detect.

In order to extract a sufficiently high number of 2D-3D correspondences for camera parameter estimation, the patterns used in our approach consist of a regular grid of rectangles that are arranged in rows and columns as shown in Fig. 3.1. The distance between adjacent regions is given as input parameter d_{region}. Furthermore, the length of the side of a square region is equal to d_{region}. The number of rows and columns in the grid as well as the overall size of the patterns can be chosen depending on the application. The number of rows and columns $n_{\text{grid},x}$ and $n_{\text{grid},y}$, respectively, is given as another input parameter. To define and detect the orientation of a pattern, we use a specific region in one corner of the pattern that merges three squares into an L-shaped region. All patterns have a unique identifier that allows them to be distinguished from each other. The identifier is encoded in one row of rectangles of the pattern. All other rows consist of regions of square shape.

The identifier is a binary coded number where squares represent 0s and rectangles represent 1s. The rectangles are twice as long and half as thick as the squares resulting in the same area as the squares. The grid of squares, including the coded identifier and the marker, are surrounded by a frame. Using such a calibration pattern, provides enough 2D-3D correspondences for the estimation process.

Our method is designed to handle images of several distributed calibration patterns. Hence, a pattern is or can be imaged multiple times by the same camera as well as by other cameras. The unique identifier and the information which image has been captured by which camera allows to generate a "connection graph". This connection graph is constructed iteratively and contains the visibility information between cameras and imaged patterns.

Using this connection graph and the input images of the calibration patterns correspondences for the parameter estimation are generated. In a step-wise manner those correspondences are used to finally estimate camera and pattern parameters.

3.1.2 Correspondence Points and Pattern Identification

In the first step we generate correspondence points and identify each visible calibration pattern. In order to determine the position of the rectangular and square regions of the calibration pattern different image processing methods are performed. For more details, see [81, 57, 146]. For each input image I_{input} these are following steps:

1. The input image I_{input} is converted to a luminance image I_{lum} [80]:

$$I_{\text{lum}} = 0.299 \cdot I_{\text{input},r} + 0.587 \cdot I_{\text{input},g} + 0.114 \cdot I_{\text{input},b}, \qquad \forall \mathbf{x} \in I_{\text{input}} \tag{3.1}$$

   where I_{lum} denotes the pixel of the image I_{lum} and $I_{\text{input},r}, I_{\text{input},g}, I_{\text{input},b}$ denote the red, green, and blue channel of the image I_{input}.

2. We classify each pixel in the luminance image I_{lum} individually as black or white. To do this, we threshold the image with a specified threshold t resulting in a binary image I_{bin}:

$$I_{\text{bin}} = \begin{cases} 1, & I_{\text{lum}} > t, \\ 0, & \text{else}, \end{cases} \qquad \forall \mathbf{x} \in I_{\text{lum}}. \tag{3.2}$$

3. In order to remove noise in the image I_{bin} we perform a morphological opening followed by a morphological closing. The binary image I_{bin} is filtered with the same

structure element M. As a result of this step, we obtain the filtered, binary image I_{filtered}

$$I_{\text{filtered}} = (I_{\text{bin}} \circ M) \bullet M. \tag{3.3}$$

As structuring element M we use a centered square of size $2 \cdot t_M + 1 \times 2 \cdot t_M + 1$ with parameter t_M.

4. A pattern is identified by segmenting the filtered, binary image I_{filtered} and counting the number of distinct segments that each segment is adjacent to. Since the number of regions in each row and column is known, the total number of distinct segments a calibration pattern consists of, is known as well.

5. The 2D-3D correspondences of a calibration pattern are established: The 2D positions are obtained by computing the center points of each rectangular and square region. From the L-shaped marker in each calibration pattern its orientation is determined. From the orientation and the given distances of the regions between each other the absolute 3D positions are computed. In our case we assume the regions to lie in the x-y-plane. Hence, the 3D position has z-coordinate 0.

6. Last, the binary encoded number in the pattern has to be identified. This is done by regarding the pixels of a region as a distribution around the region center. By performing a PCA as described in Eqn. 2.32, the eigenvalues and eigenvectors for each region can be computed. We compare the ratio of the two eigenvalues and consider the directions of the two eigenvectors. This allows to identify each region either as squares or rectangle, since all but the first row of the pattern consist of squares only.

3.1.3 Generation of Connection Graph

In the next step we assign globally consistent, unique identifiers to all patterns and we generate a connection graph.

First, all occurrences of the same calibration pattern in multiple images taken by the cameras have to be addressed. Because the position of the pattern or the cameras might have changed a new, unique identifier is used for the same calibration pattern for each captured image. This new, unique identifier is consistently applied to the corresponding pattern for all the camera views. This way one calibration pattern can be used to generate more correspondence points and increase the robustness of the calibration result.

Second, the cases have to be handled in which a calibration pattern is not seen in all cameras. To address this problem we introduce a data structure that we refer to as connection graph. It is the bipartite, undirected graph in which cameras and patterns are represented as nodes. Whenever a pattern is visible in a camera view, we add an edge in the graph connecting the corresponding camera and pattern node. The added edge means that position and orientation of the camera can be estimated with respect to the pattern.

This approach is based on the following two ideas: On the one hand, if two patterns are visible in one image, the position and orientation between those patterns can be estimated (see Sec. 3.1.4: single view alignment). On the other hand, if one pattern is visible in two different camera views, the position and orientation between the two cameras can be estimated (see Sec. 3.1.4: multiple view alignment).

The position and orientation of a pattern relative to the camera can be estimated, if (and only if) there is a path from the corresponding camera node to the corresponding pattern node in the connection graph. Globally consistent camera and pattern parameters can be estimated if (and only if) the connection graph is connected.

3.1.4 Parameter Estimation

Having generated the connection graph, we will now show how consistent camera and pattern parameters are estimated for the scene. For simplicity, in the following we will describe the problems as if the scene is observed by multiple static cameras. However, a single moving camera or multiple moving cameras can be handled in the same way by just generating a new virtual static camera for each point in time.

The estimation of camera and pattern parameters is done in six steps:

1. Estimation of camera parameters for each pattern in each camera view using Tsai's approach.

2. Alignment of all patterns visible in each camera view.

3. Estimation of pattern parameters and intrinsic camera parameters for all patterns visible in each camera view.

4. Consistent alignment of all cameras and all patterns.

5. Estimation of consistent camera and pattern parameters for all patterns and cameras.

6. Refinement of 2D points and re-estimation of camera parameters (optional).

In our approach we represent cameras as described in Eqns. 2.12 and 2.13. We denote the projection matrix corresponding to camera k by $\mathtt{P}_k$. The orientation of the camera is represented by the three Euler angles φ, ϑ, and ρ as described in Eqn. 2.18. The calibration pattern l is represented as the 4×4 transformation matrix from local to global coordinate system given in Eqn. 2.5. The pattern parameters consist of the three Euler angles α, β, and γ and the position of the pattern $\underline{\mathbf{D}} \in \mathbb{R}^3$. The three Euler angles form a 3×3 orthogonal rotation matrix $\mathtt{S}$ and the transformation matrix for the pattern is denoted by $\mathtt{Q}_l$.

The projection of a 3D point $\mathbf{X}$ of pattern l given in homogeneous coordinates in the pattern coordinate system into the camera view k is then given by

$$\mathbf{x}_{k,l} = \mathtt{P}_k \cdot \mathtt{Q}_l \cdot \mathbf{X}, \tag{3.4}$$

or explicitly written as

$$\mathbf{x}_{k,l} = \mathtt{K}_k \begin{bmatrix} \mathtt{I}_{3\times3} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathtt{R}_k^\top & -\mathtt{R}_k^\top \cdot \underline{\mathbf{C}}_k \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathtt{S}_l & \underline{\mathbf{D}}_l \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}, \tag{3.5}$$

where $\mathbf{x}_{k,l}$ is the corresponding 2D point in the image plane of camera k given in homogeneous coordinates. Following the naming given in Sec. 2.1.3 we have the calibration matrix $\mathtt{K}_k$, the rotation matrix $\mathtt{R}_k$, and the center $\underline{\mathbf{C}}_k$ of camera k. Furthermore, $\mathtt{S}_l$ is the rotation matrix and $\underline{\mathbf{D}}_l$ the position of calibration pattern l.

Lens distortion is modeled as in Eqns. 2.19 and 2.20. Hence, we only model radial lens distortion since it is the dominant distortion in the lens. Furthermore, we use the parameters k_1 and k_2 in our distortion model.

For the sake of clarity, we denote intermediate results $\mathtt{P}$ and $\mathtt{Q}$ of the camera and pattern parameter estimation of the corresponding processing step m with an additional index: $\mathtt{P}^{(m)}$ and $\mathtt{Q}^{(m)}$.

Tsai Calibration

The first step in the estimation process is to estimate camera parameters for each calibration pattern individually. This step is performed after the calibration patterns have been identified and assigned globally consistent identifiers in the case of a calibration pattern being imaged multiple times.

We estimate the camera parameters $\hat{\mathtt{P}}_{k,l}^{(1)}$ for camera k with respect to the calibration

pattern l using the established correspondences between the measured 2D points $\tilde{\mathbf{x}}_{j,k,l}$ in the image plane and the true 3D points $\bar{\mathbf{X}}_j$ on the calibration pattern. The camera parameters are computed using Tsai's method [161], as described in Sec. 2.1.5. More specifically, we estimate the focal lengths f, the rotation matrices $\mathtt{R}$, and the positions $\mathbf{C}$ of the cameras.

Furthermore, we perform a bundle adjustment (see Eqn. 2.55) to minimize the re-projection error:

$$\underset{\hat{\mathtt{P}}_{k,l}^{(1)}}{\operatorname{argmin}} \sum_j d\Big(\tilde{\mathbf{x}}_{j,k,l},\, \hat{\mathtt{P}}_{k,l}^{(1)} \bar{\mathbf{X}}_j\Big)^2 \qquad \forall k, l, \tag{3.6}$$

where $d(\mathbf{a}, \mathbf{b}) = \sqrt{||\mathbf{a} - \mathbf{b}||}$ denotes the Euclidean distance. In this step we assume furthermore, that the calibration pattern l lies in the x-y-plane of the world coordinate system. Hence, the pattern parameters $\bar{\mathtt{Q}}_{k,l}^{(1)}$ are given by $\bar{\mathtt{Q}}_{k,l}^{(1)} = \mathtt{I}_{4\times4}$.

Single View Alignment

Given the estimated camera parameters $\hat{\mathtt{P}}_{k,l}^{(1)}$ from the first step, we align the camera parameters $\hat{\mathtt{P}}_{k,l}^{(1)}$ from each camera. We obtain consistent extrinsic camera parameters and at the same time the relative positions and orientations of pattern. To do this, we move all patterns $\bar{\mathtt{Q}}_{k,l}^{(1)}$ into the corresponding camera coordinate system by

$$\hat{\mathtt{Q}}_{k,l}^{(2)} := \begin{bmatrix} \hat{\mathtt{S}}_{k,l}^{(2)} & \hat{\underline{\mathbf{D}}}_{k,l}^{(2)} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \text{with } \hat{\mathtt{S}}_{k,l}^{(2)} = \hat{\mathtt{R}}_{k,l}^{(1)\top} \text{ and } \hat{\underline{\mathbf{D}}}_{k,l}^{(2)} = -\hat{\mathtt{R}}_{k,l}^{(1)\top} \cdot \hat{\underline{\mathbf{C}}}_{k,l}^{(1)}, \qquad \forall k, l, \tag{3.7}$$

$$\hat{\mathtt{P}}_{k,l}^{(2)} := \hat{\mathtt{K}}_{k,l}^{(2)} \begin{bmatrix} \mathtt{I}_{3\times3} & \mathbf{0} \end{bmatrix} \mathtt{I}_{4\times4}, \qquad \forall k, l. \tag{3.8}$$

Note, that from the two equations above the following holds:

$$\hat{\mathtt{P}}_{k,l}^{(1)} \bar{\mathtt{Q}}_{k,l}^{(1)} \bar{\mathbf{X}}_j^{(1)} = \hat{\mathbf{x}}_{j,k,l}^{(1)} = \hat{\mathbf{x}}_{j,k,l}^{(2)} = \hat{\mathtt{P}}_{k,l}^{(2)} \hat{\mathtt{Q}}_{k,l}^{(2)} \bar{\mathbf{X}}_j^{(2)} \qquad \forall j, k, l. \tag{3.9}$$

Single View Bundle Adjustment

In the third step we incrementally perform a bundle adjustment (see Eqn. 2.55) for the camera parameters $\hat{\mathtt{P}}_{k,l}^{(2)}$ and the pattern parameters $\hat{\mathtt{Q}}_{k,l}^{(2)}$ similar to an incremental bundle adjustment.Let n_k be the number of calibration patterns that are visible in camera k. Iteratively we compute:

$$\underset{\hat{\mathtt{P}}_k^{(3,i)} \hat{\mathtt{Q}}_{k,l}^{(3,i)}}{\operatorname{argmin}} \sum_{j,l \leqslant i} d\Big(\tilde{\mathbf{x}}_{j,k,l},\, \hat{\mathtt{P}}_k^{(3,i-1)} \hat{\mathtt{Q}}_{k,l}^{(3,i-1)} \mathbf{X}_j\Big)^2 \qquad \text{for } i = 1, \ldots, n_k, \forall k, \tag{3.10}$$

with initialization $\hat{\mathtt{P}}_k^{(3,0)} := \hat{\mathtt{P}}_k^{(2)}$ and $\hat{\mathtt{Q}}_{k,l}^{(3,0)} := \hat{\mathtt{Q}}_{k,l}^{(2)}$ and results $\hat{\mathtt{P}}_k^{(3)} := \hat{\mathtt{P}}_k^{(3,n_k)}$ and $\hat{\mathtt{Q}}_{k,l}^{(3)} := \hat{\mathtt{Q}}_{k,l}^{(3,n_k)}$ for all calibration patterns l.

Multiple View Alignment

In the forth step we align all cameras and patterns with respect to the first camera. Using the generated connection graph we are able to select two camera views k and k' that have a pattern l in common.

The camera k' is aligned to camera k by applying the transformation $\mathtt{T}_{k,k'}$, which consists of the inverse transformation corresponding to pattern l in view k' followed by the transformation corresponding to pattern k in view k:

$$\mathtt{T}_{k,k'} := \hat{\mathtt{Q}}_{k,l}^{(3)} \left(\hat{\mathtt{Q}}_{k',l}^{(3)} \right)^{-1}. \tag{3.11}$$

The transformation $\mathtt{T}_{k,k'}$ is applied to the camera $\hat{\mathtt{P}}_{k'}^{(3)}$ and all pattern $\hat{\mathtt{Q}}_{l',k'}^{(3)}$ that have not been already aligned:

$$\hat{\mathtt{P}}_{k'}^{(4)} := \hat{\mathtt{P}}_{k'}^{(3)} \, \mathtt{T}_{k,k'}^{-1}, \tag{3.12}$$

$$\hat{\mathtt{Q}}_{l}^{(4)} := \mathtt{T}_{k,k'} \hat{\mathtt{Q}}_{k',l}^{(3)}. \tag{3.13}$$

For initialization, the first camera $\hat{\mathtt{P}}_1^{(3)}$ and all patterns $\hat{\mathtt{Q}}_l^{(3)} := \hat{\mathtt{Q}}_{1,l}^{(3)}$ visible in this camera view are used. By iteratively selecting cameras k' in which an aligned calibration pattern is visible and aligning that camera k' as described above, consistent camera and pattern parameters are obtained. Aligning all cameras k to camera 1 is possible, if the connection graph is connected.

Multiple View Bundle Adjustment

After the multiple view alignment, we perform more bundle adjustments (see Eqn. 2.55) to minimize the re-projection error of the patterns $\hat{\mathtt{Q}}_l^{(5)}$ in all camera views $\hat{\mathtt{P}}_k^{(5)}$. As in the Single View Bundle Adjustment, we optimize the camera parameters and the pattern parameters in this step. Iteratively we compute:

$$\underset{\hat{\mathtt{P}}_k^{(5,i)} \hat{\mathtt{Q}}_l^{(5,i)}}{\operatorname{argmin}} \sum_{j,k \leqslant i,l} d\Big(\tilde{\mathbf{x}}_{j,k,l}, \, \hat{\mathtt{P}}_k^{(5,i-1)} \hat{\mathtt{Q}}_l^{(5,i-1)} \mathbf{X}_j \Big)^2 \qquad \text{for } i = 1, \ldots, k. \tag{3.14}$$

As in the single view bundle adjustment step, for initialization of these bundle adjustment steps $\hat{\mathtt{P}}_k^{(5,0)} := \hat{\mathtt{P}}_k^{(4)}$ and $\hat{\mathtt{Q}}_l^{(5,0)} := \hat{\mathtt{Q}}_l^{(4)}$ are used. The results are $\hat{\mathtt{P}}_k^{(5)} := \hat{\mathtt{P}}_k^{(5,k)}$ and $\hat{\mathtt{Q}}_l^{(5)} := \hat{\mathtt{Q}}_l^{(5,k)}$.

Refinement

As an optional sixth step a 2D point refinement is performed. The 2D points $\tilde{\mathbf{x}}_{j,k,l}$ of calibration pattern l in camera k have been computed as the center (of area) of the projected rectangular and square regions. Because this center is not invariant under perspective projection, the measured 2D point does not correspond to the projection of the true geometric center (of area) of the region. With the estimated parameters $\hat{\mathtt{P}}_k^{(5)}$ and $\hat{\mathtt{Q}}_l^{(5)}$ for all cameras k and all calibration patterns l we can correct the measured 2D points.

A 2D offset vector is used to compensate for the error of the 2D point $\tilde{\mathbf{x}}_{j,k,l}$ and finally, to obtain the corrected 2D point $\tilde{\mathbf{x}}'_{j,k,l}$. This 2D offset vector is computed as the difference between the projected geometric center (of area) of the region and the center (of area) of the projected region.

The projected geometric center of a region is obtained by the intersection of the two diagonals of that region. It is invariant under perspective projection. The projected region appears as a quadrilateral in the image. A square or rectangular region is convex, hence the projected region is convex as well. The center of a convex quadrilateral can be computed as follows: First, compute the centers of the four triangles a quadrilateral can be split into. These four centers form another quadrilateral. Then, compute the intersection of the two diagonals of that new quadrilateral.

Using the new 2D points $\tilde{\mathbf{x}}'_{j,k,l}$ we re-estimate all parameters:

$$\underset{\hat{\mathtt{P}}_k^{(6)}\hat{\mathtt{Q}}_l^{(6)}}{\operatorname{argmin}} \sum_{j,k\leqslant i,l} d\Big(\tilde{\mathbf{x}}'_{j,k,l},\, \hat{\mathtt{P}}_k^{(6)}\hat{\mathtt{Q}}_l^{(6)}\mathbf{X}_j\Big)^2. \tag{3.15}$$

We initialize the bundle adjustment with the results from the previous bundle adjustment: $\hat{\mathtt{P}}_k^{(6)} := \hat{\mathtt{P}}_k^{(5)}$ for all k and $\hat{\mathtt{Q}}_l^{(6)} := \hat{\mathtt{Q}}_l^{(5)}$ for all l.

Our approach is flexible: Given a size of the patterns the size of the rectangular and square regions can be traded for the number of rows and columns in the pattern. On the one hand bigger regions lead to a more accurate detection of their centers. On the other hand, increasing the size of the regions reduces their number. Hence, by the choice of the size of the regions accuracy in detecting them can be traded for the number of correspondences. However, several images can be captured of the same pattern from approximately the same position in order to increase the number of correspondences.

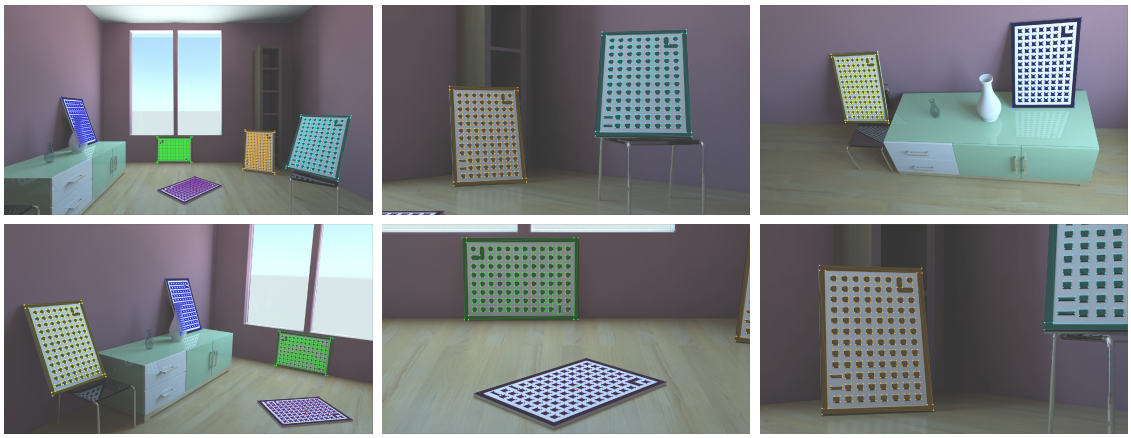

Figure 3.2: Synthetic image sequence. 6 out of 9 images shown. The positions of the projections of the estimated patterns into the image planes of the estimated cameras can be seen.

Additionally, the calibration patterns do not have to be visible in all the images. Our method can detect whether it is possible to obtain parameters for all cameras and patterns. If this is not possible, the method can compute parameters for the distinct sets of cameras and patterns that are visible in the input images.

Furthermore, the number of correspondences can be simply increased by capturing more images of the calibration patterns. By distributing several calibration patterns in the scene the volume can also be sampled in a better way. Hence, a better calibration accuracy is obtained within this volume.

3.1.5 Results

This section presents results of tests performed to evaluate our approach. In a first test, we used synthetic data to be able to compare the estimated camera parameters with the ground truth. In a second test, we took images from several calibration patterns located on scale paper to compare estimated pattern positions with measured pattern positions. Finally, we applied our method to calibrate two cameras of a robotic head which can perform human-like movements.

Synthetic Ground Truth Example

We rendered a series of 9 images of a scene in which we placed six calibration patterns (see Fig. 3.2 and 3.3). Generating a synthetic series of images enabled us to compare our

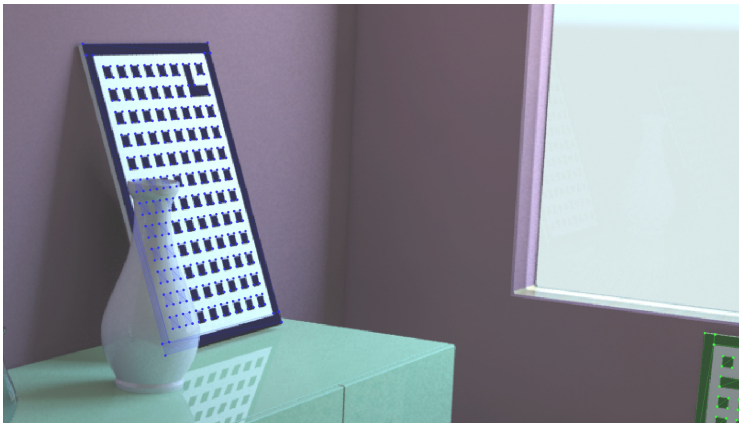

Figure 3.3: Synthetic image sequence. Detail of one of the images. Due to occlusion this pattern is not visible in the image, however, from the other images the position of the pattern can be estimated accurately.

Image	Error		RMSE
	f [mm]	$\mathbf{C}$ [mm]	φ, ϑ, ρ $\times 10^{-4}$ [rad]
1	0.00861	0.07769	1.8215
2	0.06058	0.26194	9.3651
3	0.01667	0.10513	7.0942
4	0.00287	0.03357	2.5241
5	0.00593	0.11377	1.1414
6	0.00130	0.13067	1.1775
7	0.00637	0.06130	0.9941
8	0.00428	0.07432	8.3503
9	0.05864	0.11363	11.946
all	0.02936	0.12170	6.3581

Table 3.1: Synthetic image sequence. Comparison between estimated camera parameters and ground truth without 2D point refinement. Errors given for focal length, camera center and RMSE for Euler angles of orientation.

Image	Error		RMSE
	f [mm]	$\mathbf{C}$ [mm]	φ, ϑ, ρ $\times 10^{-4}$ [rad]
1	0.00166	0.01962	1.7127
2	0.01492	0.05603	7.9734
3	0.00311	0.02640	6.3498
4	0.00106	0.01355	2.2049
5	0.00872	0.05367	1.1054
6	0.00238	0.03902	1.2050
7	0.00026	0.01868	1.0056
8	0.00161	0.03465	6.7190
9	0.06134	0.07239	11.350
all	0.02852	0.10208	5.6700

Table 3.2: Synthetic image sequence. Comparison between estimated camera parameters and ground truth with 2D point refinement. Errors given for focal length, camera center and RMSE for Euler angles of orientation.

estimation results with the ground truth (see Tab. 3.1 and 3.2). To put the accuracy of the estimation results into relation, the rendered virtual room has a size of approximately 4×4 meters. Compared to the overall extend of the scene, the observed errors can be regarded as very low.

We performed two runs of camera parameter estimation using our approach. In the first run we estimated camera and pattern parameters *without* refining the 2D points obtained from Sec. 3.1.2, while in the second run we refined the 2D points. Although we found the estimation results using the initial 2D points to be good, they could be improved by the 2D point refinement (an improvement of approx. 3%, 16%, and 11% percent for focal length, camera center and camera rotation, respectively).

Real-world Ground Truth Example

For a real-world example we printed seven calibration patterns and arranged them on a paper with millimeter scale. Twelve images of this scene were taken. Since we placed the pattern on the scale paper, in this example all patterns lay in one plane. By using scale paper we were able to measure the corners of the patterns. From the corner points we then computed the centers of the patterns. After applying our camera calibration approach, we were able to compare the estimated pattern positions to the measured ones (see Tab. 3.3).

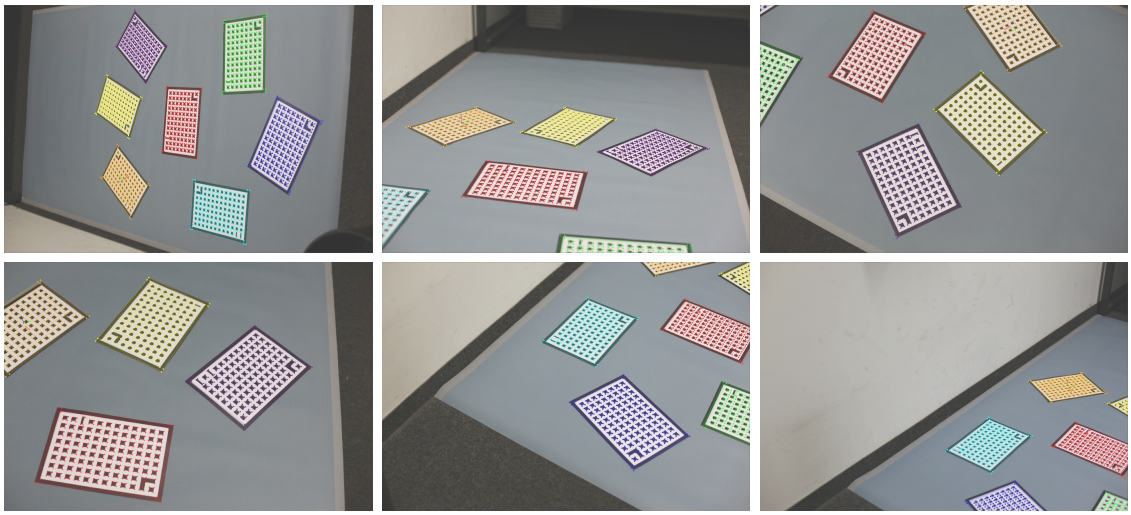

Figure 3.4: Real-world image sequence: 6 out of 12 images shown. Patterns placed on paper with millimeter scale. Projections of estimated patterns into the camera images are overlaid.

The largest absolute difference in Tab. 3.3 is 1.14 mm in relation to a 583 mm absolute pattern distance (corresponding to a relative deviation of 0.2%).

3.2 Photometric Calibration

The geometric calibration estimates the position and orientation of the cameras in the world coordinate system and the intrinsic camera parameters that govern the projection. In addition to this geometric calibration a photometric calibration of the cameras has to be performed as well. This calibration step relates the measured intensities of the camera in the acquired image to the physical intensities in the real world.

This calibration step consists of two parts:

1. *Color mapping* or *color balancing* refers to the process of mapping the measured colors in the image to true colors. To do that, a color chart is used [31]. The color checker is a chart consisting of several different patches with calibrated colors (see Fig. 3.5). Capturing an image of such a color checker allows to compute how the true color components red, green, and blue, $\bar{r}, \bar{g}$, and $\bar{b}$ are mapped to the imaged color

—	876.94	354.41	368.27	477.54	823.98	829.50
876.94	—	714.22	711.34	399.43	285.58	333.02
354.41	714.22	—	568.29	382.47	559.94	807.66
368.27	711.34	568.29	—	380.13	784.79	540.83
477.54	399.43	382.47	380.13	—	406.04	425.21
823.98	285.58	559.94	784.79	406.04	—	582.95
829.50	333.02	807.66	540.83	425.21	582.95	—
—	876.63	354.20	368.03	477.58	823.16	829.40
876.63	—	713.65	711.24	399.09	285.16	332.37
354.20	713.65	—	567.77	382.19	559.11	806.94
368.03	711.24	567.78	—	380.14	784.00	541.16
477.58	399.09	382.19	380.14	—	405.20	424.76
823.16	285.16	559.11	784.00	405.20	—	581.81
829.40	332.37	806.94	541.16	424.76	581.81	—
—	0.30	0.21	0.23	0.04	0.82	0.11
0.30	—	0.58	0.10	0.34	0.42	0.65
0.21	0.58	—	0.51	0.28	0.83	0.72
0.23	0.10	0.51	—	0.01	0.80	0.33
0.04	0.34	0.28	0.01	—	0.83	0.44
0.82	0.42	0.83	0.80	0.83	—	1.14
0.11	0.65	0.72	0.33	0.44	1.14	—

Table 3.3: Real-world evaluation results. All values are given in millimeters. Distance between pattern i and j is given in column i and row j. **Top**: Measured distances between pattern centers. **Middle**: Distances between estimated pattern positions. **Bottom**: Difference between top and middle table.

components $\tilde{r}, \tilde{g}$, and $\tilde{b}$. This allows to compute a 3×3 matrix $\mathtt{C}$

$$\begin{pmatrix} \tilde{\bar{r}} \\ \tilde{\bar{g}} \\ \tilde{\bar{b}} \end{pmatrix} = \mathtt{C} \cdot \begin{pmatrix} \bar{r} \\ \bar{g} \\ \bar{b} \end{pmatrix} \tag{3.16}$$

by a least squares method (see Sec. 2.2). The measured and reference color values are normalized to the interval $[0, 1]$ before computation of $\mathtt{C}$. With this matrix $\mathtt{C}$ the colors of all subsequently taken images can be balanced by applying the inverse matrix $\mathtt{C}^{-1}$:

$$\begin{pmatrix} \bar{r} \\ \bar{g} \\ \bar{b} \end{pmatrix} = \mathtt{C}^{-1} \cdot \begin{pmatrix} \tilde{\bar{r}} \\ \tilde{\bar{g}} \\ \tilde{\bar{b}} \end{pmatrix}. \tag{3.17}$$

This way the true colors of the captured object or scene are obtained independent of the lighting conditions. Performing this color balancing explicitly also performs a white balancing. For only performing white balancing, a gray balance card as shown in Fig. 3.6 is already sufficient. However, using the 3×3 matrix $\mathtt{C}$ for color balancing captured images using a color checker only works if the camera sensor has a linear response. Hence, in addition a response curve estimation has to be performed as follows.

2. *Response curve estimation* is the task of estimating the response of the camera's sensor to different amounts of incoming light. The obtained response curve relates the amount of incoming light and the measured pixel values of the sensor. In our pipeline the response curve has to be estimated in order to determine whether the camera sensor has a linear response or not. With a linear response, the measured pixel values are proportional to the amount of incoming light and hence the exposure time. Camera sensors are designed to have a linear response [155], however, other steps of the image processing pipeline of the camera's digital signal processor (DSP) can introduce non-linearities (see Sec. 3.2.1). Photographic film in analog cameras has also a non-linear response [82]. For estimating the response curve a color checker is used. In this case a sequence of images with different exposures is captured. Since the true color values of the patches of the color checker and the exposure times for each image are known, a response curve for each color component can be fitted to the measured data using least squares techniques (see Sec. 2.2). The measured

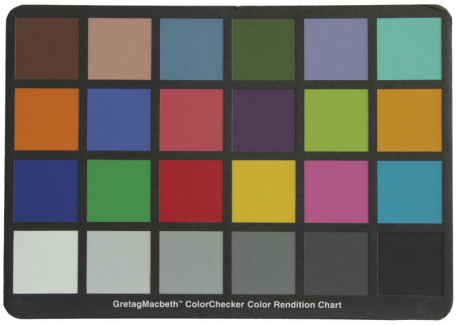

Figure 3.5: Color Rendition Chart

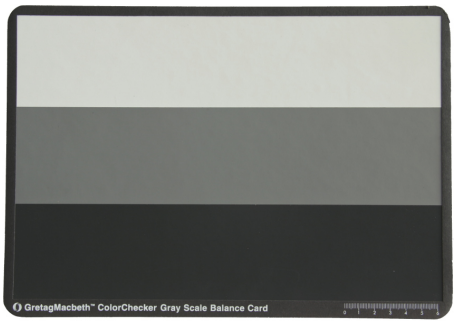

Figure 3.6: Gray Scale Balance Card

and reference color values are normalized to the interval $[0, 1]$ before response curve estimation.

For more in-depth information about color mapping see [66, 31]. For camera response curve functions and their estimation we refer to [114, 60].

3.2.1 Camera Image Processing Pipeline

Different processing steps are performed by the camera's DSP during the development of the raw image to a JPG image. We will briefly explain the reasons why those processing steps are applied to the raw image. They include:

- Setting the black and saturation level: Because of the photo-electric effect the CCD or CMOS sensor of the camera measure in each sensor cell a voltage that depends on the amount of incoming photons. Then these voltages are amplified depending on the ISO setting that corresponds to the film speed, which is the sensitivity of a photographic film. Finally, the amplified voltages are converted by an analog-to-digital converter. The black level corresponds to the voltage that a sensor cell measures, if no photons reach the cell. The saturation level is the measured voltage when a cell is saturated (see Fig. 4.3). The voltage of a saturated sensor cell cannot increase, even if more photons hit the cell. Hence, the range that the sensor can measure is determined by the black and saturation level. It is mapped to the range $[0, 2^{14} - 1]$.

- Noise reduction: There are different sources for noise in the measuring process [66]. The most significant one is the noise introduces by the ISO setting of the camera. The voltage that is measured in the sensor cells due to the photo-electric effect is

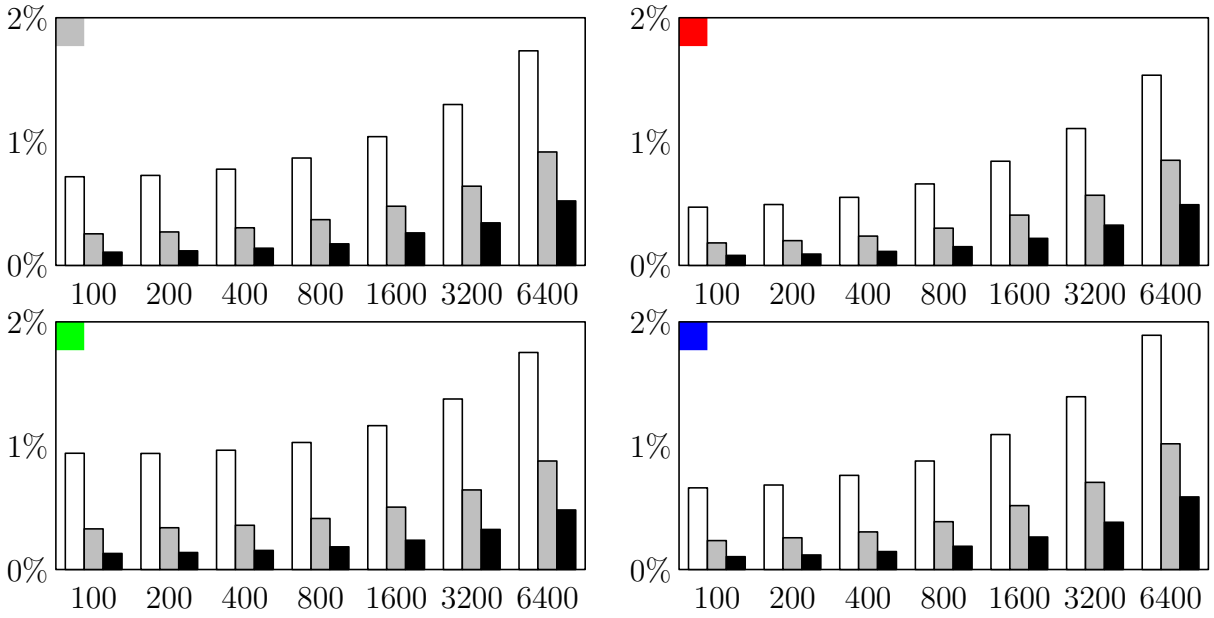

Figure 3.7: Noise in the images with different ISO settings ranging from ISO 100 to ISO 6400. Standard deviation of the white, gray, and black patch of the gray scale balance card shown in Fig. 3.6. Values averaged over 10 images. Standard deviation given relative to range of values. From **top left** to **bottom right**: Noise of luminance computed from RGB, Noise in red channel, green channel, and blue channel.

amplified before being converted by the ADC. The ISO setting of the camera controls this amplification. It corresponds to the sensitivity of the photographic film in analog cameras. Lower ISO settings lead to less amplification and reduced sensitivity, while high ISO settings lead to more amplification and increased sensitivity. However, amplifying the measured voltage introduces additional noise. The higher the ISO setting, the more noise is introduced, which has to be dealt with in a noise reduction step (see Fig. 3.7).

- White balancing: The appearance of individual colors in the image depends on the color spectrum of the illumination which is given as a color temperature. White balancing refers to the task of correcting the acquired colors' appearances in the image to a "natural" appearance. Here, natural appearance refers to the appearance in daylight, which corresponds to a temperature of 5500 K. White balancing is done either automatically, or by correcting the specified temperature of the light sources.

- Demosaicing (also called "debayering"): The CCD or CMOS sensor of digital cameras cannot measure different colors. In order to generate RGB images red, green, and

blue filters in front of the sensor cells are used. The filters for all three color channels are typically arranged in a repeating pattern for each 2×2 block of sensor cells. This pattern is also called *Bayer pattern*. The pattern consists of a red filter in one corner, a blue filter in the opposite corner, and green filters for the two remaining corners of each 2×2 block. The Bayer pattern only yields one color component per pixel. The interpolation process that fills in the missing information for the other color channels per pixel is called *demosaicing* or *debayering*.

- Image sharpening: Because of the Bayer pattern in the raw image a demosaicing algorithm has to be applied. The demosaicing process interpolates the measured and fills in the missing color information. In this way RGB values for every pixel are generated. The interpolation process can be thought of as a down-sampling of the image from full resolution to half resolution, followed by an up-sampling back to full resolution. This is comparable to a smoothing of the original image. Hence, by demosaicing a smoothed image is generated. In order to compensate for the smoothing an image sharpening is applied.

- Gamma correction: The mapping of linear input values from the camera's sensor to output values is called *gamma correction*. The input values are mapped exponentially with a given parameter γ. This step is performed in order to give an image the same appearance as in reality: The human visual system has a non-linear, logarithmic response. Hence, in order for the image to appear as in reality the gamma correction maps the measured values accordingly.

- Rescaling: CCD and CMOS sensors have limited color depth. Our cameras have 14 bits color depth, but consumer DSLR cameras with 12 bits are also often encountered. However, 8 bits is the color depth that all common file formats support, while 16 bits color depth is supported by some file formats as well. Hence, in order to store the acquired information the data with 12 or 14 bits of color depth has to be scaled to fin in the range of the file format. Additionally, the measured values in all the RGGB Bayer pattern blocks can be corrected by scaling the red, the blue, and the two green channels separately.

- Image compression: In order to use less space on the hard drive or faster transmission over network, the processed image is stored using the JPG file format. JPG is a format with lossy compression. The steps that are being performed in the conversion include color space conversion, chroma sub-sampling, discrete cosine transform,

quantization, and entropy encoding. Depending on the quality setting more or less information is lost when converting the image to JPG. Hence, after conversion the original image cannot be retrieved anymore. On the other hand, in comparison to the loss in visual image quality the size of the image is significantly reduced.

For most of the steps above different methods and/or different settings exists. The final JPG image is one from many possible developed images. However, the majority of the steps are designed to create a JPG image that can be viewed by the photographer to assess for example the quality of the image, the composition of the motive, or the exposure.

Chapter 4

Acquisition Hardware

This chapter describes the acquisition hardware that we use in our pipeline. First, we describe the illumination hardware that we use to generate controlled illumination. Second, we describe the camera setup and setting that we use for capturing the objects in the light stage. For precise 3D reconstruction it is important both, to generate accurate illumination and correctly process the acquired data in order to obtain physically meaningful measurements. They are crucial for the computation of normal maps, which is finally presented in the last section.

4.1 Light Stage

The light stage is an apparatus that allows us to generate controlled illumination. It is a sphere-like structure. At its inside light emitting diodes (LEDs) are attached. Their brightness can be controlled individually to generate different illuminations. We use the Light stage to generate specific illuminations that allows us to compute normal maps. The normal maps are then used as input to our 3D Reconstruction method presented in Sec. 5.

4.1.1 Structure

The structure of the light stage itself is a geodesic dome. Geodesic domes are three dimensional polyhedra that resemble spheres. One possibility for obtaining a geodesic dome such as the light stage that we use, is to construct it in the following way: The basic shape of the geodesic dome is a regular icosahedron. Since the regular icosahedron is a Platonic solid, a circumscribed sphere exists that passes through all vertices of the icosahedron.

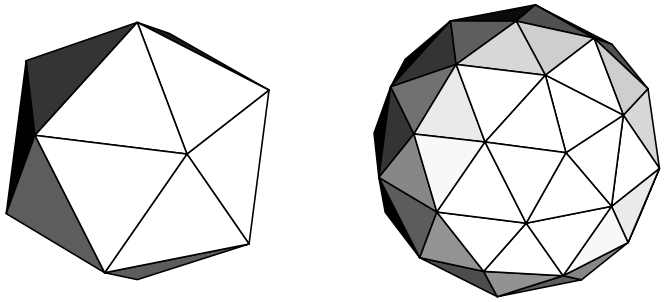

Figure 4.1: The structure of the light stage. **Left**: The regular icosahedron is the basic shape of the light stage. **Right**: The geodesic dome that forms the shape of the light stage. The dome is obtained from subdividing each triangle of the icosahedron into four triangles. LEDs are placed at each vertex and in the middle of each edge.

Among the regular polyhedra made of triangles the icosahedron is the one that is most similar to a sphere. It consists of 20 equilateral triangles, 12 vertices and 30 edges. By subdividing each of its triangles into smaller triangles we obtain another polyhedron. The triangles can be subdivided into quadratically many, smaller, equally sized, equilateral triangles. Then, for this obtained polyhedron the newly introduced vertices can be moved outwards such that the circumscribed sphere passes through them as well. We obtain the structure of the light stage as shown in Fig. 4.1. Note however, that the resulting triangles are not equilateral any more.

In the case of our light stage each triangle has been subdivided into four triangles. Its diameter is approximately 80 cm. The Light Stage itself is assembled from metal bars that correspond to the edges of the polyhedron. The five triangles facing the ground are missing in order for a person to stand inside the Light Stage such that the face is in the center, or to leave room for a supporting structure for objects placed in the center of the sphere.

4.1.2 Illumination

As light sources we use 156 light emitting diodes (LEDs) in total: 42 LEDs are placed at the vertex positions, 120 at the midpoints of each edge, and 6 are missing at the bottom part of the Light Stage because of the five left out triangles.

Most light sources emit a continuous electromagnetic spectrum of light. This means the light contains different wave lengths that inside and outside the visible spectrum. The visible spectrum contains light of wave lengths approximately between 380 nm and 780 nm. The different wave lengths in the spectrum have different intensities. The color spectrum of the light sources is related to the so-called *black body radiation*: A black body of a certain

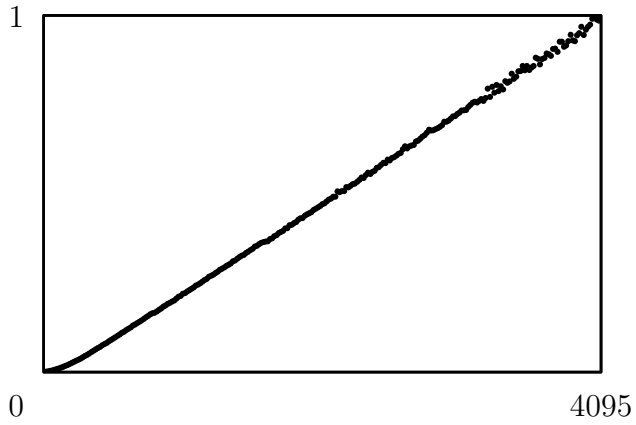

Figure 4.2: Brightness of LEDs depending on input value of LED drivers. Brightness scaled to the range $[0, 1]$.

temperature emits a characteristic electromagnetic spectrum. The color spectrum of the light source can be characterized by the corresponding temperature value given in Kelvin of a black body emitting an electromagnetic spectrum that best approximates the spectrum of the light source. Light sources that emit a continuous spectrum are, for example, the sun or incandescent light. Other light sources, like vapor lamps, emit a spectrum of light that contains only one or a few characteristic wave lengths.

The LEDs attached to the Light Stage emit a spectrum that can be characterized by approximately 6000 K color temperature. The brightness is individually controlled by pulse-width modulation by LED drivers in 4096 steps between 0 and 4095, which corresponds to 12 bit of information. Furthermore, the brightness of the LEDs can be controlled linearly as shown in Fig. 4.2. The drivers use the serial peripheral interface (SPI). The SPI is a serial interface designed for synchronized serial data transmission. Each driver can control 16 light emitting diodes, so in order to control all light sources of the Light Stage 10 drivers are daisy-chained and controlled by a micro-controller. In our case the brightness values for all light sources are sent as a data sequence of 156 12-bit values by the micro-controller to the drivers. The chaining of the drivers results in the first driver setting the brightness of the first 16 light sources and forwarding the remaining data sequence to the next driver. The second driver then sets the brightness of the next 16 light sources and forwards the remaining data sequence to the next driver and so on, until the brightness of all LEDs have been set.

4.1.3 Micro-controller

We use an Arduino ATMega1280 micro-controller with 16 MHz clock frequency that can handle 54 digital input/outputs, 16 analog inputs and can operate one SPI. The micro-controller has 128 KB of flash memory, where approximately 124 KB can be used for storing the code of uploaded programs. Furthermore, 8 KB of RAM memory are available to the executed program.

Programs for the Arduino micro-controller are written in a restricted version of the C programming language. These restrictions of the programming language come from the limited abilities of the processor. A integrated development environment as well as plug-ins for developing programs for the Arduino are available. They contain libraries that allow to read from, write to, and set inputs and outputs. Furthermore, a compiler is available, as well as an uploader that allows to upload the compiled program to the micro-controller via USB connection.

As mentioned above the serial peripheral interface is used to transmit the sequence of brightness values for all LEDs to the LED drivers. From the constructions of the Light Stage as a regular icosahedron with subdivided triangles the positions of the individual light sources can be computed. Those positions are stored in the flash memory of the micro-controller to be accessible during program execution.

The micro-controller is also used to release the cameras' shutters. This is done in the following way: one digital output pin of the Arduino activates a electro-mechanical relay that closes a larger circuit containing up to ten relays. Each of those relays is connected to one camera via the remote control terminal. This remote control terminal allows both, activating the auto-focus and releasing the shutter. We do not use the ability of the remote control terminal to auto-focus, because auto-focus takes some additional time that reduces the maximal number of images we can capture per second. Furthermore, by using auto-focus before every image acquisition it is not guaranteed that the camera will focus on the same distance between two image acquisitions. Hence, we use the functionality of releasing the shutter through the circuitry only.

The delay introduced by one relay is 5 ms according to the specifications. Hence, the delay of the entire circuitry for releasing the camera shutters is about 10 ms. On the other hand, the exposure time that we use is at least around 40 ms (1/25). Additionally, the cameras introduce a delay between the signal of releasing the shutter and the image acquisition. Another delay is introduced after the image acquisition when the captured information is read out from the CMOS sensor. Those delays result in a maximal frame

rate of the camera of about 3.7 frames per second (fps), which corresponds to about 270 ms. So, the delay introduced by the circuits with relays is low compared to the delay of the cameras and the exposure time that is used for image acquisition. Hence, from these figures and from measurements we assume that the image captured are synchronized in the sense.

4.2 Cameras

In our hardware set-up we use identical Canon 550D consumer DSLR cameras. The cameras are mounted on tripods. The image stabilization is turned off, because otherwise we experienced misaligned images within acquired image sequences. The misalignment happens because the image stabilizer is correcting for movement that does not exist. The cameras have a CMOS sensor with a resolution of 5184×3456 pixels, corresponding to approximately 18 million pixel (18 MP). The sensor has a size of approximately 22.3×14.9 mm, which is roughly 38 % the size of a full frame sensor. Hence, the pixel size of the camera's sensor is approximately 4.3×4.3 μm. The camera's crop factor, that is the factor the focal length has to be multiplied with to obtain a comparable field of view using a full frame sensor, is 1.6.

The camera uses a 14-bit analog-to-digital converter (ADC): The analog voltage measured by the sensor is converted to a digital value with 14 bits precision. Hence, the camera sensor can distinguish 2^{14} different shades of brightness. All cameras are equipped with an identical Canon EF-S 18–135 mm f/3.5–5.6 IS zoom lens, that allows to adjust the effective focal length of the lens between 18 and 135 mm with an aperture of f/3.5 at 18 mm and f/5.6 at 135 mm.

4.2.1 Acquisition settings

All images are captured in the manufacturer's raw file format CR2 (Canon original RAW 2nd edition) and in the JPG file format at the same time. As with every image given in a raw file format, several settings can be changed after the image has been acquired and hence, the image can be edited to some extent before it is developed and saved as a JPG file. We acquire images with aperture f/7.1, exposure time of 1/40 s and an ISO setting of 800. The focal length of the cameras' lenses is set to approximately 32 mm, which corresponds to a focal length of 50 mm for full frame cameras, because of the crop factor of 1.6. The aperture represents a compromise between the good depth of field and a sufficient amount of incoming light in order to reduce the shutter time. The ISO setting represent a

good compromise between noise in the image and sensor sensitivity in order to reduce the shutter time for a general acquisition set-up (see Fig. 3.7).

However, in case of static objects the acquisition time is less important than in case of acquiring for example moving objects, a human face or a facial expression. With static objects the shutter time can be increased in order to compensate a smaller aperture which increases the depth of field. Furthermore, a lower ISO setting can be used to reduce noise in the images.

The acquired image in the JPG file format is an image that is developed by the camera's digital signal processor (DSP) from the corresponding raw image with the given camera settings at acquisition time (see Sec. 3.2.1).

However, we are interested in using the cameras to obtain physically correct measurements of the incoming light. Hence, we want to minimize the post-processing to a minimum and only perform the necessary development steps as described in the following section. That means, we are only interested in setting and correcting for the black and saturation level of the raw images. In order to have full control of the other processing steps we handle them ourselves.

4.2.2 Image Development

In this section we present how we process the raw image data. This include the usage of tools for developing the raw images as well as the post-processing that we use for further developing the image. The main focus is to obtain physically correct measurements of the incoming light. From estimating the camera response curves as described in Sec. 3.2 we know that the cameras in our setup have linear response. The estimated response curves are presented in Fig. 4.3 and 4.4. With the knowledge of the camera sensor having linear response, we obtain easily interpretable results: A measured value of 0 corresponds to no incoming light, higher values indicate higher amounts of incoming light, where for example doubling the amount of incoming light also results in doubled values.

We use the tool "dcraw" [33], that is freely available for Windows and UNIX/Linux operating systems. This tool enables us to convert the raw images acquired by the cameras from the proprietary CR2 file format of the cameras into the file format PPM. The PPM (Portable Pixmap) file format has several advantages: In contrast to CR2, the majority of image viewers can display PPM files without any additional plug-ins. Second, PPM is a simple uncompressed file format, which can be processed and converted easily into other file formats, like PNG. The PNG file format is a good candidate for archiving files, because

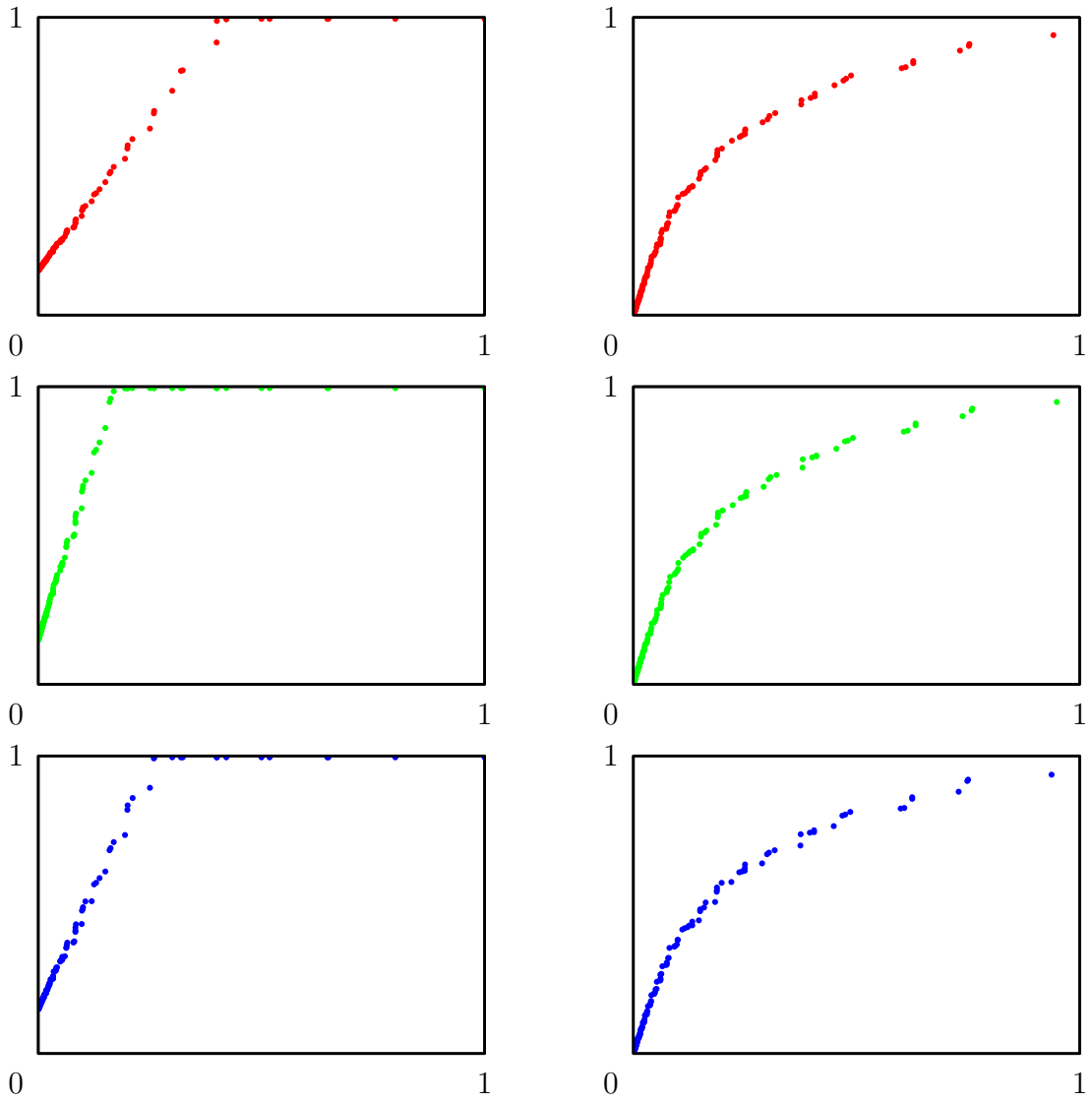

Figure 4.3: Response curves for red, green, and blue channel of images of color chart. Response values scaled to range $[0, 1]$. **Left**: Raw response curves as acquired by the camera sensor and stored in raw file format (CR2). **Right**: Response curves obtained from images developed by the camera's DSP and stored in JPG file format. Black and saturation level adjusted.

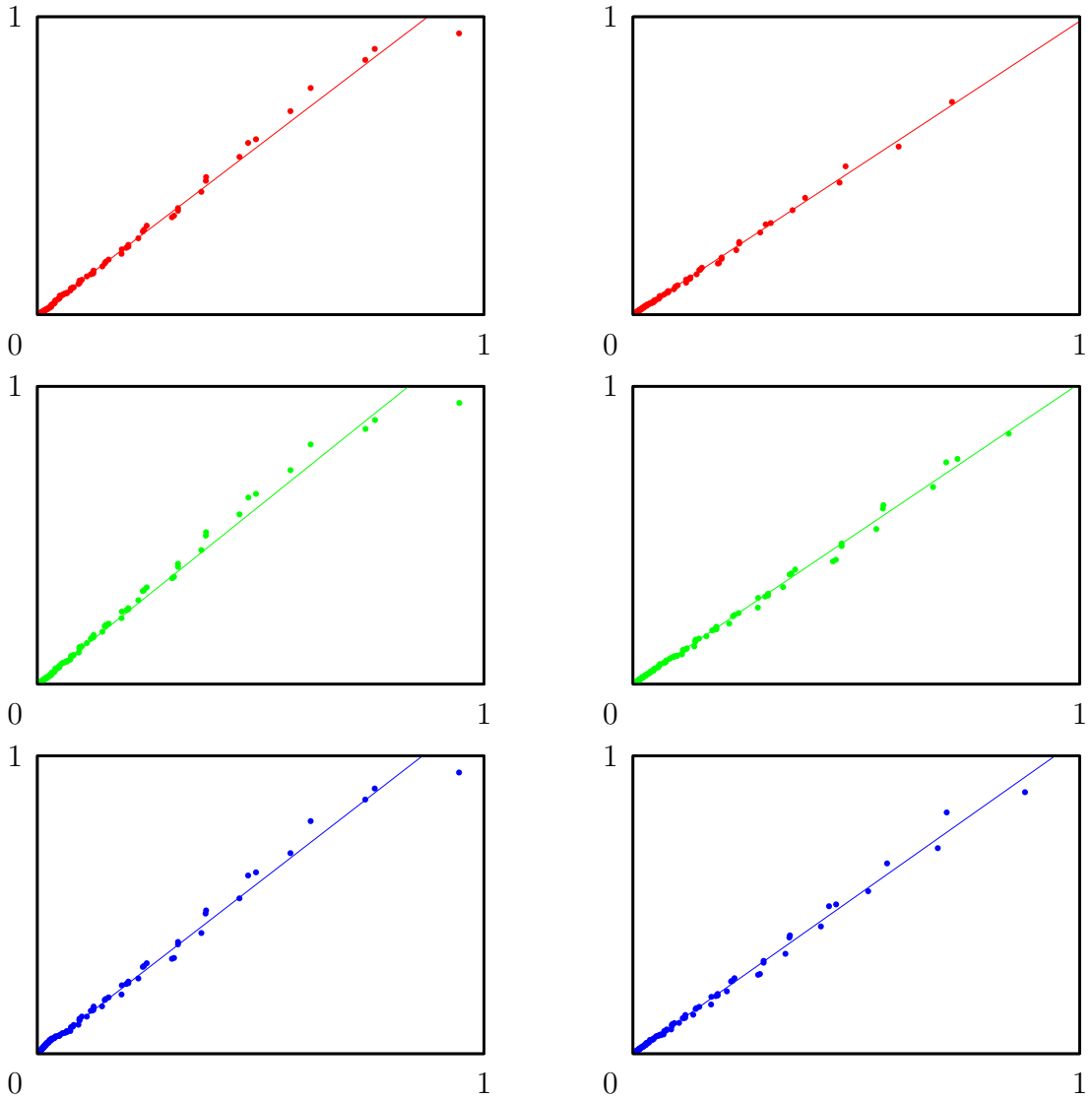

Figure 4.4: Response curves for red, green, and blue channel of images of color chart. Radiance and response values scaled to range $[0, 1]$. Linear response function fitted to response curves. **Left**: Raw images converted with Canon Digital Photo Professional software. **Right**: Raw images converted with DCRAW software as used in our pipeline.

it allows lossless compression, which minimizes file sizes while preserving the original image quality. Furthermore, we also use "dcraw" to automatically set the dark and saturation level of the raw input images.

We decided to use "dcraw" instead of the software supplied with the cameras by the camera manufacturer. The camera manufacturer provides both, a separate application and a software development kit. Both have the same functionality. We prefer "dcraw" to those two options, because they do not give full control over the raw image processing pipeline or the methods and parameters used in it. Furthermore, "dcraw" can be integrated into an automatic processing pipeline much easier: No additional program using the development kit has to be written in order to convert the raw images from CR2 to a common image file format.

We use "dcraw" to do the following: Read and convert the raw image from CR2 to PPM file format; adjust the dark and saturation levels of the raw image; and finally, scale the raw values to a 16-bit range.

In a second step we demosaic the converted image the following way: We generate an image that is half the size of the original image in both dimensions. The values of the red and blue channel of a 2×2 Bayer pattern block are mapped directly to the red and blue component of the RGB image. The values of both green channels are averaged before being mapped to the green component of the RGB image. This leads to an image of half the size, since the values of the four cells of a 2×2 Bayer pattern block are mapped to the red, green, and blue value of one pixel in the output image. This method is also known as *binning*. With binning no interpolation method has to be used to fill in the missing data.

Demosaiced images have the same resolution as the camera sensor, but red, green, and blue information in each pixel, in contrast to the raw image. This is because demosaicing interpolates the values.

However, from an information theoretic point of view the amount of information does not increase by the interpolation, only the amount of data does. In this sense, by mapping the four values of a 2×2 Bayer pattern cell to one RGB value with three components, we reduce the amount of information by $\frac{1}{4}$. On the other hand, because of averaging the two green channels, the obtained green component of the RGB value is measured twice as accurate as the red or blue component.

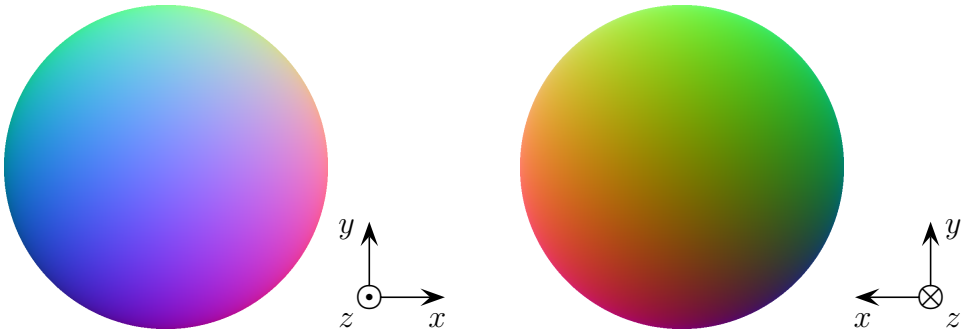

Figure 4.5: Color coding of the normals of a sphere in the normal maps. **Left**: Normals of a hemisphere with view in negative z-axis direction. **Right**: Normals of the other hemisphere with view in positive z-axis direction.

4.3 Normal Maps

We use the Light Stage to generate normal maps. Normal maps are images that contain information about the normal vector of the captured surface instead of color or brightness information. The normal vectors of the captured surface are color coded in the normal map as shown in Fig. 4.5. In this section we will explain how we obtain normal maps using the Light Stage and the set-up of cameras.

4.3.1 Gradient Illumination

In Eqns. 2.74 – 2.79 we have seen that the x-component of the surface's normal at any point is proportional to the irradiance in the case of light coming from the direction of the positive x-axis. In the same way, the y- and z-component of the normal are proportional to the irradiance in the case of light coming from the positive y- and z-axis, respectively.

With the Light Stage we can generate illuminations that approximates light coming from one direction. In fact, the illuminations generated by the Light Stage are not restricted to coincide with one coordinate axis: Illuminations from any direction can be generated.

This is done as follows: First, the light stage coordinate system has to be aligned to the coordinate system in which the cameras have been calibrated. We achieve this by physically aligning one calibration pattern with respect to the light stage during the acquisition of the image sequence for calibration.

From the construction of the light stage as described in Sec. 4.1.1, the positions of the light sources are known. We simply assume that the light stage itself coincides with a sphere in 3D space. The coordinate system is defined in such a way that both, the x- and

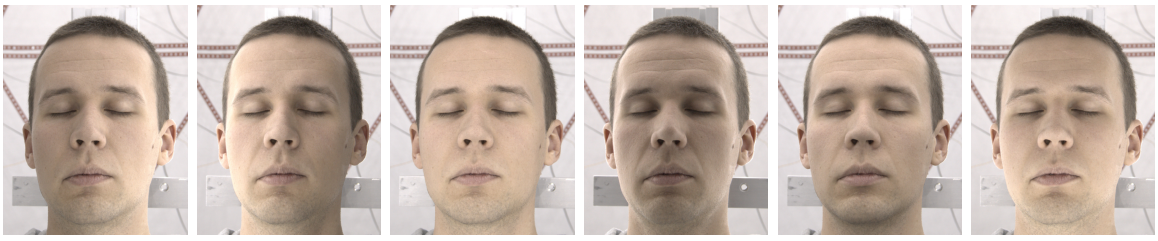

Figure 4.6: Gradient illuminations generated in the light stage. **Left** to **right**: Illumination from the right (I_x), left (I_{-x}), front (I_z), back (I_{-z}), top (I_y), and bottom (I_{-y}).

y-axis are parallel to the ground and each of both axes is parallel to one wall. This leaves the z-axis to be perpendicular to the ground. The cameras are also calibrated with respect to this coordinate system. In this coordinate system the normalized direction $\mathbf{d}_{LED}$ of emitted light of an LED is given by

$$\mathbf{d}_{LED} = -\frac{\mathbf{p}_{LED}}{||\mathbf{p}_{LED}||}, \tag{4.1}$$

where $\mathbf{p}_{LED}$ is the position of the LED. When we want to generate an illumination with light coming from direction $\mathbf{i}$ the brightness b_{LED} is computed as the scalar product between the light direction $\mathbf{l}$ and the direction of light emission $\mathbf{d}_{LED}$ of the LED:

$$b_{LED} = \mathbf{l}^\top \mathbf{d}_{LED}. \tag{4.2}$$

However, this way also negative values for b_{LED} can occur. Since negative light cannot be emitted in reality the values of b_{LED} in the range $[-1, 1]$ are mapped linearly to the range $[0, 1]$ by computing b_{LED} as

$$b_{LED} = \frac{1}{2} \cdot \mathbf{l}^\top \cdot \mathbf{d}_{LED} + \frac{1}{2}. \tag{4.3}$$

4.3.2 Normal Map Computation

In order to retrieve the x-, y-, and z-component of the normal in the easiest and fastest way, the illumination patterns should coincide with the directions of the x-, y-, and z-axis. One way to that is to use three illumination patterns with light directions $(-1, 0, 0)^\top$, $(0, -1, 0)^\top$, and $(0, 0, -1)^\top$ representing light coming from the positive x-, y, and z-axis.

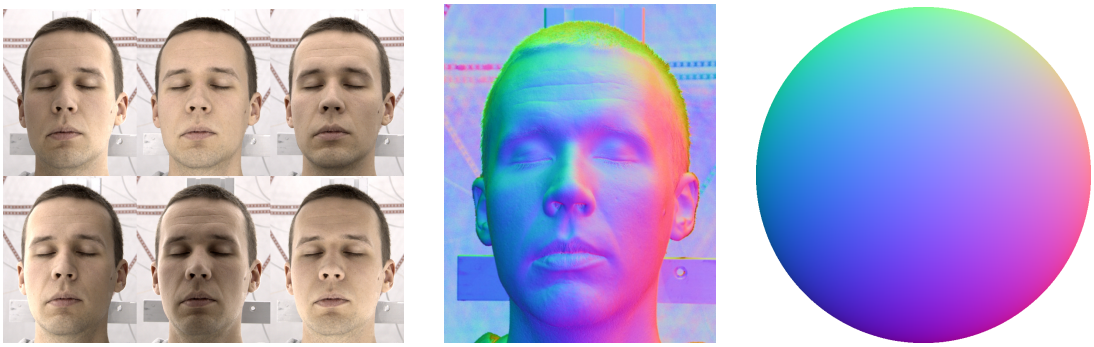

Figure 4.7: From **Left** to **right**: Input images captured in light stage with gradient illuminations, normal map computed from input images, reference color coding of normals.

In Addition, a forth illumination with constant illumination

$$b_{LED} = \frac{1}{2} \tag{4.4}$$

for all light sources has to be used to compute the normal $\mathbf{n}$ as

$$\mathbf{n} = \frac{(I_x - I_d, I_y - I_d, I_z - I_d)^\top}{||(I_x - I_d, I_y - I_d, I_z - I_d)^\top||}, \tag{4.5}$$

where I_d is the radiance of a pixel under constant illumination and I_x, I_y, and I_z are the radiance under illuminations from x-, y-, and z-direction [107].

Another way of computing the normal is to used six illuminations [164]: Additionally to the three illuminations I_x, I_y, and I_z the three illuminations I_{-x}, I_{-y}, and I_{-z} with light directions $(1,0,0)^\top$, $(0,1,0)^\top$, and $(0,0,1)^\top$ are used (see Fig. 4.6). In those illuminations the light is coming from the negative x-, y-, and z-direction. Then, the normal $\mathbf{n}$ is computed via

$$\mathbf{n} = \frac{(I_x - I_{-x}, I_y - I_{-y}, I_z - I_{-z})^\top}{||(I_x - I_{-x}, I_y - I_{-y}, I_z - I_{-z})^\top||}, \tag{4.6}$$

Computing the normals as described above has several advantages. An image with light direction $(-1,0,0)^\top$ behaves complementary to an image with light direction $(1,0,0)^\top$. The same applies for other light directions. That means bright parts in one image appear dark in the other one and vice versa. Using two such complementary images for each component of the normal increases the accuracy and robustness of the computation. Furthermore, computing the normals as presented in Eqn. 4.6 makes them invariant under linear transformations.

The computed normals are stored as an RGB image. The continuous range $[-1, 1]$ for each component of the normal $\mathbf{n} = (n_x, n_y, n_z)^\top$ is mapped to the discrete range of $[0, 65535]$ in case of a 16-bit image:

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \left(\mathbf{n} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \cdot 32767.5 = \left(\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \cdot 32767.5. \tag{4.7}$$

The values n_x, n_y, and n_z are stored in the red, green, and blue channel r, g, and b, respectively. As an example two normal maps of a unit sphere are given in Fig. 4.5: One normal map for the front hemisphere and another one for the back hemisphere. In Fig. 4.7 the normal map of a face is shown as an example.

The normal $\mathbf{n}$ is reconstructed from the red, green, and blue channel r, g, and b of the normal map as follows:

$$\mathbf{n} = \frac{(r - 32767.5, g - 32767.5, b - 32767.5)^\top}{||(r - 32767.5, g - 32767.5, b - 32767.5)^\top||}. \tag{4.8}$$

Chapter 5

3D Reconstruction

3D reconstructions of mid-sized real objects are often obtained by triangulation methods. In this area one can differentiate between active and passive methods.

Active triangulation methods, such as laser scanning [98] or structured-light-scanning [131], modify the captured scene by projecting a high-frequency time-varying illumination pattern onto the object. The accuracy is typically limited by the resolution of the projected patterns. These methods are currently used the most in practical applications, because of their high robustness. However, the capturing speed is limited by the projector and acquisition hardware. Furthermore, for a complete reconstruction the object must be captured from multiple viewpoints. Because of interferences of the illumination patterns, acquisition from multiple viewpoints cannot be performed in parallel, which makes the overall process slow.

In contrast, passive triangulation methods, such as stereo [136] or multi-view-stereo [138], rely solely on two or more images taken from different viewpoints. The robustness of passive stereo strongly depends on matchable image features. Therefore, a passive stereo reconstruction requires a surface reconstruction method to obtain a closed surface. The main advantages of passive triangulation methods are their reduced hardware effort because no projectors are required and their speed because multiple viewpoints can be captured in parallel.

Both, passive and active triangulation methods can be combined with photometric stereo. Photometric stereo [166, 107] is a technique that allows reconstructing high-resolution surface normals by observing the shading of an object under different lighting conditions from a single viewpoint. In contrast to active triangulation methods, the illuminations are not high-frequency patterns but rather multiple distributed low-frequency

illuminations that allow to invert the shading model in order to estimate the surface normal.

In this section we propose a novel, flexible method that can be classified as a combination of passive triangulation and photometric stereo [58]. We generate a time-varying spherical gradient illumination and observe the scene with multiple cameras. Detailed normal maps of the inspected object from different viewpoints are obtained using photometric stereo. The normal maps are the input to our multi-view stereo algorithm which generates a 3D reconstruction. Then the normal information is employed to interpolate between the sparsely sampled stereo estimates in order to obtain a high-resolution reconstruction.

In contrast to most existing approaches, we employ the normal information not solely to refine the reconstruction. Instead, the normal information is used in the matching process of our multi-view stereo approach. Our approach enables us to use significantly larger windows during patch matching, which strongly increases its robustness. As a consequence, smoothness constraints that are typically required in passive stereo are not necessary. Consequently, without smoothness constraints the surface can be sparsely evaluated which vastly reduces the computational effort compared to a densely sampled evaluation. The resulting reconstruction is sparse, but the included estimates are all reliable. Finally, the sparse reconstruction is interpolated with the normal information to obtain a dense and detailed reconstruction.

Our approach is designed to handle and reconstruct objects, that are static, smooth and diffuse. The limitation to static objects results from the limited capturing speed of the employed consumer digital single lens reflex (DSLR) cameras. We focus on diffuse objects or objects that are best approximated as diffuse. However, specular objects can be reconstructed to a certain extent as presented in the results giving an indication about the robustness of our method.

5.1 Related Work

This section reviews related work that combines passive triangulation methods with photometric stereo approaches for accurate 3D reconstruction.

Photometric Refinement Many approaches have been proposed using photometric information to improve and refine an initial geometry or surface. The initial geometry in these approaches can be obtained either by (multi-view) stereo reconstruction [34, 168, 169, 123], structure-from-motion [171, 100, 73, 132], or triangulation scanning [117, 84].

Figure 5.1: The hardware that generates different illuminations with three DSLR cameras mounted on a tripod.

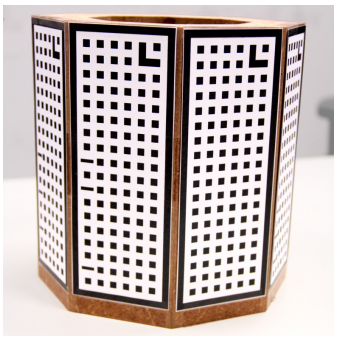

Figure 5.2: The calibration object made of individual patterns that are attached to the sides of an octagonal prism.

Other approaches employ 3D models that are morphed [170] or estimate shadow maps that are then used to reconstruct the 3D geometry [29].

In contrast to our approach, these methods have in common that normal information is not an integral part of the 3D position acquisition. Photometric information is used in a refinement step, but it is not directly employed for the generation of the initial 3D reconstruction.

Silhouettes-Based Approaches Silhouette information extracted from multiple views allows to generate a visual hull of the object. The 3D positions and normals of the visual hull can be optimized to obtain a 3D model [18, 30, 69]. The visual hull can also be used as a proxy to deform and assemble partial reconstructions to a complete 3D model [162].

However, our approach does not rely on silhouette information and also works in situations where the visual hull is not available or is not very descriptive, as for example in the case of a frontal view of a relief.

Multi-view Stereo and Normals Surface normals and positions can also be conjointly estimated using a set of images captured under multiple point light illuminations [14] by using a known example object [1]. In contrast to multi-view stereo methods our approach does not rely on detectable image features which lead to sparse point clouds where a surface has to be fitted.

Uncalibrated Photometric Stereo Furthermore, uncalibrated photometric stereo refers to the case in which the lightning conditions are not known. Different methods deal with

Figure 5.3: Generated normal maps (color-coded as RGB) for the corresponding three different viewpoints. The normals are RGB color-coded.

this scenario. They generate a 3D model of the object and can estimate the light positions [9, 46] or compensate for varying unknown illumination conditions [56].

5.2 Acquisition Setup

In order to perform a 3D reconstruction with our method, the inspected object is placed in the light stage. Images of the object under different illuminations are taken by multiple calibrated and synchronized DSLR cameras as described in Sec. 4.1. The object is captured under six different gradient illuminations as described in 4.3.1. The normal maps are computed as presented in 4.3.2 (see also [164]).

Fig. 5.1 shows the hardware that is used to generate the illuminations for photometric stereo reconstruction. Three DSLR cameras as described in Sec. 4.2 are mounted on a tripod for image acquisition. In Fig. 5.3 the generated normal maps from the corresponding viewpoints of the cameras are shown. The normals are consistent across the views, which is the prerequisite to employ the normal information as a matching score for multi-view stereo.

An accurate (geometric) camera calibration is important, because calibration errors directly propagate into the 3D reconstruction. The camera calibration estimates the relation between cameras by estimating the camera parameters. We use the approach presented in Sec. 3.1. As a calibration object we use an octagonal prism with calibration patterns attached to its sides as shown in Fig. 5.2. The calibration object is placed inside the light stage at the position where the object is placed during the acquisition.

5.3 Photometric Multi-View Stereo

In this section, we present our method for multiple view 3D reconstruction using normal maps obtained from photometric stereo. First, we explain how we obtain a sparse reconstruction. Second, we describe how a dense reconstruction is obtained from a sparse reconstruction. Last, a final filtering step of the dense reconstruction is presented.

Overview

Low frequency noise that is present in the input normal maps renders direct integration methods unsuitable to accurately reconstruct the true 3D geometry of the object [117], while high frequency noise leads to wrong reconstructions of local surface features.

Instead, in the first step (detailed in Section Sec. 5.3.1), normal information from photometric stereo is used to improve the patch matching capabilities of multi-view stereo. In this step, 3D patch surfaces are generated using normal information in a reference view. Reliable and accurate depth values are obtained by optimizing the 3D patch matching costs. However, computing the depth values for all pixels is computationally very expensive. Hence, we reconstruct the depths of a small number of points resulting in a sparse reconstruction. This reduces the computation time to a few minutes on standard PC hardware.

In the second step (detailed in Sec. 5.3.2), we use the normal map of the reference view to interpolate the sparse reconstruction keeping the sparse points fixed. This resulting reconstruction is dense and has high accuracy, however, there are some tiny disturbing peaks at the positions of the sparse 3D points because of the quantization along the line of sight.

In a third step (detailed in Sec. 5.3.3) those peaks are removed by a normal map driven filtering. Fig. 5.4 shows the output of the individual steps of our algorithm for a face. The next section explains each step in more detail.

It should be stressed that the normal map driven filtering is different from introducing a smoothness term in the dense reconstruction step. In the reconstruction step we want to compute a dense surface that approximates the measured normal maps the best. Introducing a smoothness term results in smoothing the overall reconstructed surface, which is not desirable. In contrast, the normal map driven filtering is able to smooth areas where normal map and surface contradict each other, while enhancing areas, where normal map and reconstructed surface agree.

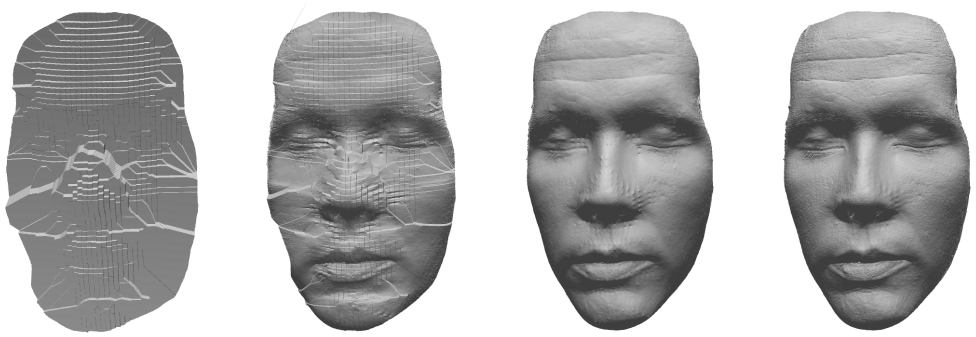

Figure 5.4: Overview of the different steps of the algorithm. **From left to right**: Initial sparse reconstruction after nearest neighbor interpolation; reconstruction obtained after 1 iteration of nonlinear least squares minimization; after 20 iterations of minimization; after 20 filtering iterations.

5.3.1 Sparse Reconstruction

In this first step an initial point cloud of the object is reconstructed that consists of reliable and accurate 3D points. For the initialization a reference view v_{ref} is set. The 3D points of the initial point cloud $\mathcal{I}$ correspond to 2D points lying on an equidistant grid in the reference view and their depths. For each grid point we use the normals given in a window of size $w \times w$ to reconstruct the local 3D geometry of that patch surface as follows.

Patch Reconstruction

Given the lines of sight $\mathbf{l}_i$, $\mathbf{l}_j$, of pixels i, j, the normal $\mathbf{n}_i$, and the candidate depth d_i of pixel i we can compute the depth d_j of pixel j as

$$d_j = \frac{\mathbf{l}_i^\top \cdot \mathbf{n}_i}{\mathbf{l}_j^\top \cdot \mathbf{n}_i} \cdot d_i. \tag{5.1}$$

Rewriting this equation yields the constraint

$$\mathbf{l}_j^\top \cdot \mathbf{n}_i \cdot d_j - \mathbf{l}_i^\top \cdot \mathbf{n}_i \cdot d_i = 0. \tag{5.2}$$

Stacking up equations for all neighbors in a 4-neighborhood of every pixel in the considered window leads to a linear system of equations

$$\mathtt{A}' \cdot \mathbf{d} = \mathbf{0}, \tag{5.3}$$

where $\mathbf{d}$ is the vector of unknown depths. Additionally, every equation (i.e., row of matrix $\mathtt{A}'$) is weighted according to a standard distribution with standard deviation σ depending on the distance of the considered pixel to the center pixel of the window.

Solving Linear System of Equations

In order to avoid the trivial solution we set the depth of the center pixel to 1 leading to the extended linear system of equations:

$$\begin{pmatrix} \mathtt{A}' \\ \mathbf{e}^\top \end{pmatrix} \cdot \mathbf{d} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \tag{5.4}$$

where $\mathbf{e}$ is the canonical unit vector corresponding to the position of the center pixel in vector $\mathbf{d}$. Choosing a different depth d_i for the center pixel i will lead to $d_i \cdot \mathbf{d}$ as solution for the over-determined linear system of equations. Writing the extended linear system of equations as

$$\mathtt{A} \cdot \mathbf{d} = \mathbf{b}, \tag{5.5}$$

we compute a least squares solution by solving the normal equation

$$\mathtt{A}^\top \cdot \mathtt{A} \cdot \mathbf{d} = \mathtt{A}^\top \cdot \mathbf{b} \tag{5.6}$$

using Choleksy decomposition (see Eqn. 2.36). The point cloud $\mathcal{P}_i$ representing the local 3D surface of that patch for pixel i is obtained by multiplying the depths with their corresponding line of sights:

$$\mathcal{P}_i := \left\{ \mathbf{X}_j | \mathbf{X}_j = \mathbf{l}_j \cdot d_j, 1 \leqslant j \leqslant w^2 \right\}. \tag{5.7}$$

Line of Sight Sampling

In order to find the 3D point $\mathbf{X}_i$ for each 2D grid pixel i, we sample depths d_i on the line of sight. We can simply transform the reconstructed patch $\mathcal{P}_i$ using $d_i \cdot \mathbf{d}$. This is a crucial advantage of our formulation because re-solving Eq. 5.5 is not required for each depth candidate. Otherwise, the sampling along the line of sight would be computationally too expensive for large patch size.

We project the reconstructed patch $\mathcal{P}_i$ from Eqn. 5.7 at depth d_i into the other views $v \neq v_{\text{ref}}$. We then compute the 3D patch matching cost $c(d_i)$ for depth d_i by comparing

Figure 5.5: Comparison of patch matching with synthetic head model from three views, patches of size 32×32 pixels used. From **left** to **right**: ground truth depth map; depth map obtained from 2D patch matching; difference between ground truth and obtained depth map (gray values adjusted for better visibility); depth map obtained from 3D patch matching; difference between ground truth and obtained depth map (gray values adjusted).

the projections of the normals in the window around the grid point with the normals at the points of projection in the other views v:

$$c(d_i) = \sum_{v \neq v_{\text{ref}}} \sum_{\mathbf{X}_j \in \mathcal{P}_i} d(N_{v_{\text{ref}}}(\mathtt{P}_{v_{\text{ref}}} \mathbf{X}_j \cdot d_i), N_v(\mathtt{P}_v \mathbf{X}_j \cdot d_i))^2, \tag{5.8}$$

where $\mathbf{X}_j$ is the j-th 3D point obtained from patch reconstruction, $\mathtt{P}_v$ is the projection matrix of camera v, and N_v denotes the normal map of view v, returning the interpolated value at the given point. We then choose the depth d_i that has the lowest cost among all depths.

A typical window size for local patch reconstruction is 160×160 pixels, which is much larger than patch sizes used by standard multi-view stereo in the image domain (which is typically 16×16 pixels or less). Increasing the patch size in the image domain is not possible because the true surface can no longer be approximated by a fronto-parallel plane as assumed by standard 2D patch matching. Fig 5.5 shows a comparison of standard 2D patch matching and our matching via local 3D patch surfaces. Because a large patch can contain much information, the cost function in Eqn. 5.8 typically has a single distinct minimum. However, if this is not the case, we discard estimates that have only small variations among similar depths:

$$\text{Var}\left[c(d_i - D_{\text{Var}}, d_i + D_{\text{Var}})\right] < t_{\text{Var}}. \tag{5.9}$$

As a consequence, our approach is highly robust. In our experiments, we observed that the reconstructed depths contain no outliers if the threshold t_{Var} was chosen correctly. As

a result of this initialization step we obtain a set $\mathcal{I}$ of grid points in the reference view v_{ref} with corresponding depths.

5.3.2 Dense Reconstruction by Nonlinear Optimization

In the second step, the set of points $\mathcal{I}$ with depths d_i from the multi-view stereo approach and the normal map $N_{v_{\text{ref}}}$ is used to reconstruct the dense surface of the object. This is done by minimizing the cost function

$$\bar{c}(\mathbf{d}) = \sum_k \sum_{j \in \mathcal{N}(k)} \left(\mathbf{l}_j^\top \cdot \mathbf{n}_i \cdot d_j - \mathbf{l}_k^\top \cdot \mathbf{n}_k \cdot d_k \right)^2 \quad \text{subject to } d_k = d_i,\ \forall i \in \mathcal{I}, \tag{5.10}$$

where $\mathcal{N}(k)$ denotes the 4-neighborhood of pixel k. Again, $\mathbf{d}$ is the vector of unknown depth, now containing all depths d_k. This cost function penalizes deviations from Eqn. 5.2, while fixing the set of 3D points $\mathcal{I}$. This is done to prevent the normals from pulling the reconstruction towards the unconstrained solution, which is known to exhibit low-frequency errors [117] on one hand and to avoid heading towards the trivial solution on the other hand. A user-defined binary mask is used to determine the region of interest which determines the set of pixels k used in the reconstruction process.

The minimization of the cost function Eqn. 5.10 is done by Newton's method as described in Sec. 2.2.2: As initial solution we use the 3D point cloud $\mathcal{I}$ of the grid points with depths from the first step. Values in between the grid are interpolated by nearest neighbor interpolation. The cost function is assumed to be locally linear and is iteratively minimized. In each iteration a linear least squares problem similar to Eqn. 5.5 is solved by Cholesky decomposition (see Eqn. 2.36). In all experiments 20 iterations have been performed.

5.3.3 Filtering

In the last step we filter the depth values obtained from minimizing Eqn. 5.10, because the depths d_i of the set of initial 3D points $\mathcal{I}$ have not been optimized. Filtering the depth values is done iteratively. Rewriting Eqn. 5.2 we obtain

$$d_j = \frac{\mathbf{l}_i^\top \cdot \mathbf{n}_i}{\mathbf{l}_j^\top \cdot \mathbf{n}_i} \cdot d_i, \tag{5.11}$$

which is used to propagate the four depth values of the four neighbors (left, right, top, and bottom neighbor) to pixel i. The average depth of the four propagated depth values is computed and updates the depth of pixel i. These depth values geometrically correspond to line-plane-intersections. For numerical stability, a propagated depth value is only used for the update, if the angle between the line of sight $\mathbf{l}_i$ and the normal $\mathbf{n}_j$ is greater than $\arccos(t_{\text{angle}})$ with threshold parameter t_{angle}. In our experiments we use $t_{\text{angle}} = 0.173$ corresponding to an angle of approximately 80°. This update is performed iteratively for all pixels. In all experiments we performed 20 iterations.

5.4 Results

In this section we first evaluate our method on synthetic data. Then we present 3D reconstructions of real objects obtained from our method. Furthermore, we compare the obtained reconstructions to results obtained from laser scanning.

5.4.1 Synthetic Data

Our method is evaluated on synthetic data. We generate a series of synthetic images of a 3D model of a human head as ground truth. We compute the normal maps for all three views. The three normal maps computed for each view and the positions of the cameras are used to reconstruct the human head. We align the ground truth head model and our reconstructions with the *iterative closest point* (ICP, [17]) algorithm. The average distance between the mesh the head model and the one of our reconstruction is used as quality measure.

We test our method in settings with different material properties: Lambertian reflectance without shadows; Lambertian reflectance with shadows and different levels of Gaussian noise added to the input images; Lambertian reflectance with shadows, specular reflections, and different levels of Gaussian noise.

The results are shown in Table 5.1: The best reconstruction is obtained in the ideal setting. With increasing level of noise the quality of the reconstruction decreases, while in general the head is reconstructed better in the absence of specular reflections (because specular reflections violate the assumption of a perfectly Lambertian surface).

	diffuse	diffuse with shadows				
Gaussian noise level [%]	0	0	1	2	3	4
AAE [%]	0.242	0.315	0.374	0.436	0.477	0.516
		diffuse and specular with shadows				
Gaussian noise level [%]		0	1	2	3	4
AAE [%]		0.402	0.434	0.468	0.528	0.544

Table 5.1: Average alignment error (AAE) of reconstructions to ground truth. The noise level is given as percentage of the range of pixel values of the input images. The error is given relative to the size of the ground truth model.

5.4.2 Real-World Data

For demonstrating the applicability of our method we show several reconstructions of real-world objects. We use three DSLR cameras as described in Sec. 4.2. Fig. 5.6 shows results of our reconstruction method for five objects: Relief, Purse, Shoe, Santa, and Vase. Additional close-up views of the 3D geometry with and without texture of Relief, Shoe, and Vase examples are shown in Fig. 5.7. Although the Vase has a specular surface, strong errors in the reconstruction are mainly visible at grazing angles, while other parts are reconstructed fairly well. This demonstrates that our approach is able to handle deviations from the assumptions to some extent. Figure 5.8 shows the results of reconstructing three faces. For all reconstructions we used a window size of 160×160 pixels. In all cases 20 iterations for minimizing Eqn. 5.10 and 20 additional iterations for filtering as described in Sec. 5.3.3 have been performed. t_{Var} has been set to 30.0. On standard PC hardware using unoptimized C++ code, the computation times are between 10 and 15 minutes depending on the model.

5.4.3 Comparison to Laser Scanning

We compare our method to 3D reconstructions of a laser scanner. Figure 5.9 shows a qualitative comparison between the scanned objects and our results. Our method is able to reconstruct finer details of the objects' surfaces. We align our 3D reconstructions and the ones obtained from laser scanning with the ICP method for a quantitative comparison. The distance between the two aligned meshes is used as a measure of quality. We obtain an average alignment error of 0.539% for the object Relief, 0.775% for the object Shoe, and

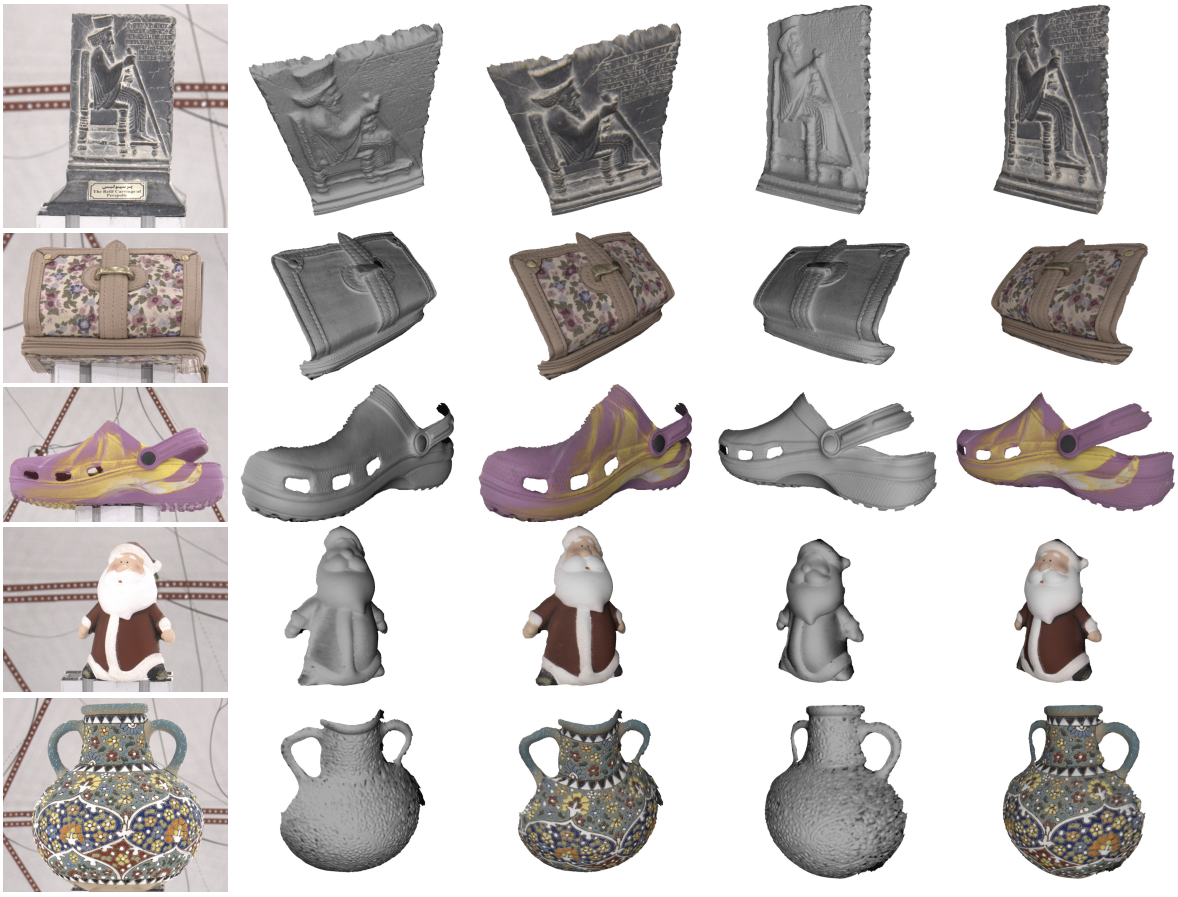

Figure 5.6: 3D Reconstructions of several objects. **Top** to **bottom**: Relief, Purse, Shoe, Santa, Vase. **Left** to **right**: Image from the reference camera; resulting 3D reconstruction shown from two different views with and without texture.

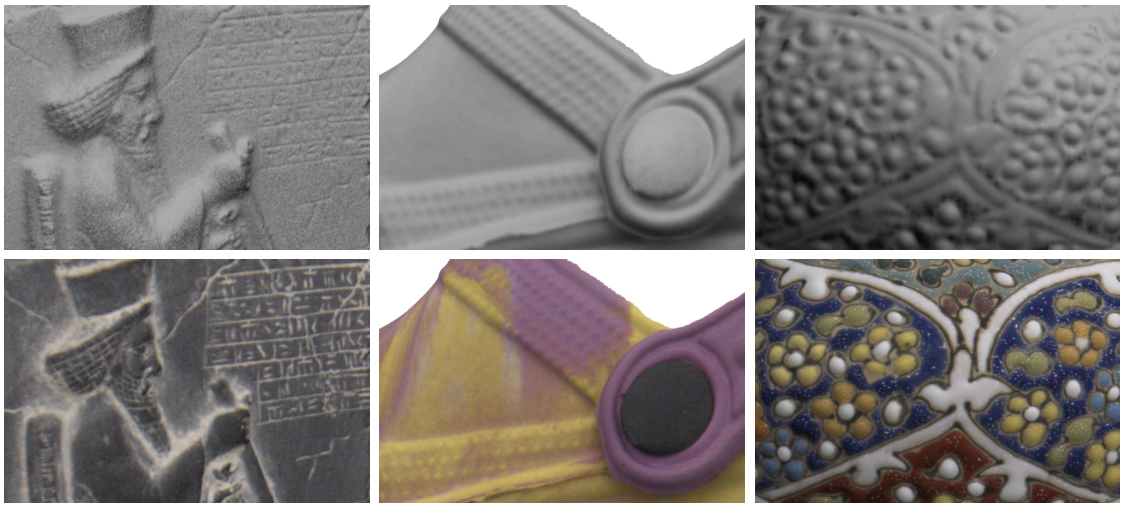

Figure 5.7: Details of the reconstructions. **Top row**; Reconstructed 3D geometry. **Bottom row**: Reconstructed 3D geometry with texture. **Left** to **right**: Relief, Shoe, Vase.

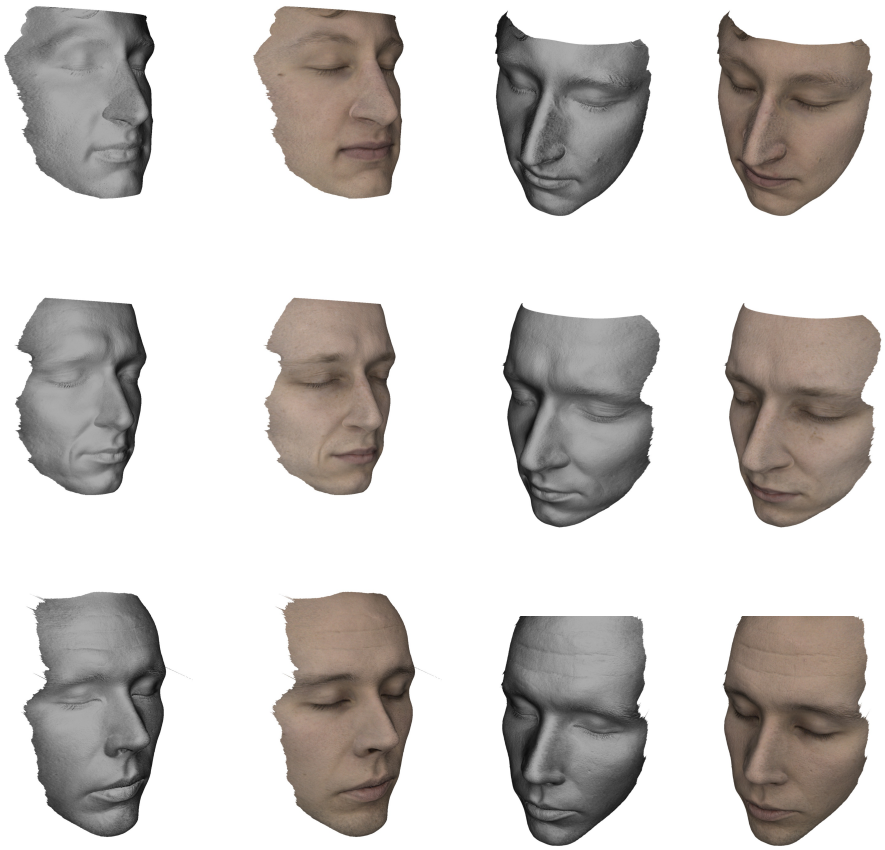

Figure 5.8: Reconstructions of three faces with 3D geometry and rendered 3D geometry with texture shown.

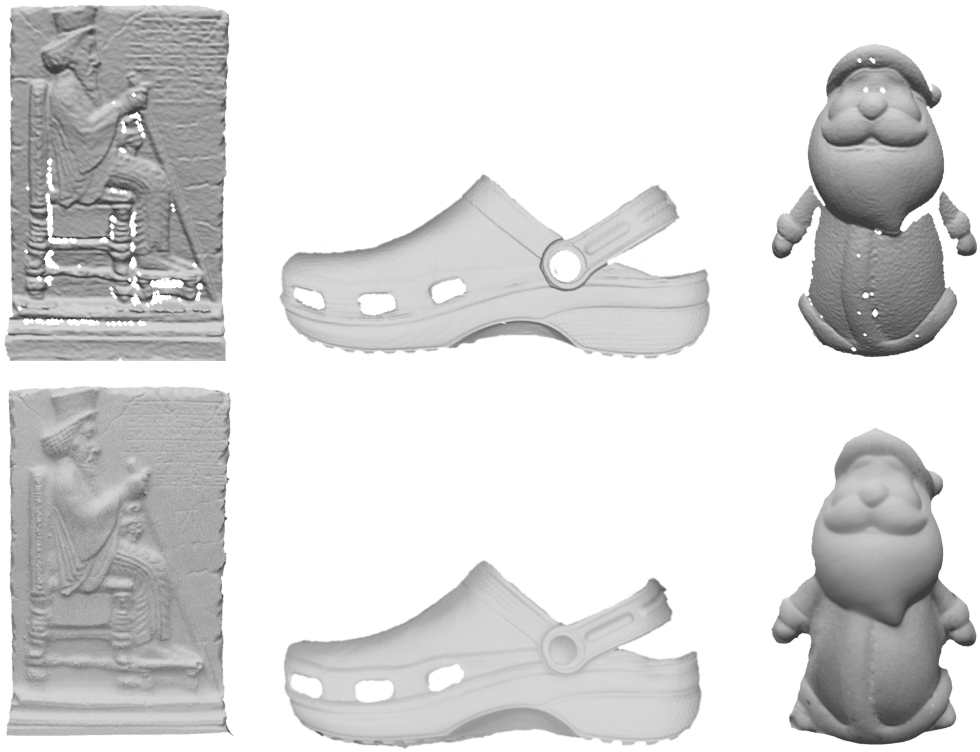

Figure 5.9: Comparison to laser scanning: Relief, Shoe, and Santa. **Top**: results from laser scanner. **Bottom**: our reconstruction result.

0.544% for the object Santa. The error is given relative to the size of the scanned object.

5.4.4 Limitations

Currently, our results are obtained only from the perspective of a single reference view. Hence, they can be parametrized in the image domain and are in fact only 2.5D (2D plus depth). Augmenting our reconstructions to more complex topologies is possible by merging multiple 2.5D reconstructions. To do that, the number of cameras must be increased in order to capture multiple 2.5D reconstructions in parallel from different view points for increased surface coverage. This does not increase the acquisition time. Merging multiple 2.5D reconstructions into a single 3D mesh can be achieved by using existing methods, such as the *volumetric range image processing* method (VRIP, [36]).

Also, we do not handle cases in which the normal maps contain significant errors which can occur at depth discontinuities, in shadowed regions, or when the material is not diffuse. This may lead to wrong reconstructions. This effect can be reduced on one hand by increasing the sampling density of the grid. In the extreme case each pixel of the grid can be sampled. In this case the dense reconstruction step as well as the filtering step can be omitted. There are no disadvantage to this approach, except for a significantly larger computational effort. On the other hand, depth discontinuities can be automatically detected. Once detected, they can be taken into account by considering neighboring pixels in Eqn. 5.10, that do not lie at depth discontinuities. Furthermore, once a 3D mesh is recovered, shadowed regions can be detected (compare with [65]) and used to improve the generated normal maps.

In theory, our approach is capable of capturing dynamic scenes. However, the employed consumer DSLR cameras are difficult to synchronize if operated at their maximum speed. In future, we would like to augment our capturing setup with more professional high-speed high-resolution cameras to obtain dynamic reconstructions.

Chapter 6

Conclusion and Future Work

We have presented a pipeline for dense 3D reconstruction using active illumination. The pipeline consists of calibration of the acquisition hardware, image development and processing, and a novel 3D reconstruction method.

The flexible camera calibration method allows accurate calibration of cameras in challenging scenarios. This includes calibration of cameras that are facing each other or that are facing away from each other. For increased accuracy of the camera parameter estimation the number of acquired images of the calibration patterns can be increased, which increases the number of point correspondences. Furthermore, the calibration estimates the positions and orientations of the patterns. Besides considering the estimation error, this allows to further assess the quality of the calibration, if the true positions and orientations of the patterns are known. By aligning a pattern to the light stage, the cameras are calibrated with respect to the coordinate system of the light stage.

The light stage is used in order to generate illuminations that are used for photometric stereo reconstruction. The acquired images are developed and processed in order to obtain physically reliable measurements. This allows to compute robust surface normals of high quality. The computed surface normals are stored as normal maps that are the input for the 3D reconstruction method.

The 3D reconstruction approach is based on photometric normals given in multiple views. The computed normal maps together with RGB images of the captured objects in multiple views are used as input. The reconstruction method uses those information to obtain an initial sparse 3D reconstruction similar to multi-view reconstruction. The normal information are then used to compute a dense 3D surface based on the initial sparse reconstruction. Finally, a filtering of the obtained surface is performed. Reconstructions

generated with this method provide high detail, because of the photometric normals and high accuracy, because of the initial multi-view reconstruction.

The contribution of this thesis is:

- A new camera calibration method for consistent calibration of multiple cameras. This method can handle difficult calibration tasks, for example cameras facing each other, showing its flexibility and robustness.

- A description of the hardware setup and image processing and development. Obtaining high quality reconstruction results relies on high quality input data. This requires special care that has to be taken during data acquisition and data processing.

- A novel method for dense, accurate, fully automatic 3D reconstruction. The method employs photometric stereo reconstruction which allows to obtain the same level of detail, while at the same time relaxing restrictions during acquisition This allows the acquisition of the object from arbitrary many view points at the same time.

The presented setup can be easily extended by addition of more cameras. The calibration procedure is designed to account for complex arrangements of cameras, such as positioning cameras around the object. Furthermore, the acquisition time is independent of the number of cameras, because the gradient illuminations for obtaining photometric normals do not interfere with each other.

As future work, the presented set-up can be extended by more cameras that are arranged around the light stage for reconstructing a full model. This way the person or object can be captured from all sides. Then, the partial reconstructions have to be merged to a single full 3D model.

Furthermore, depth discontinuities have to be automatically detected. In principle, this can be done by considering the cost function used in the initial sparse reconstruction step. The knowledge of depth discontinuities can be easily included into the dense reconstruction step.

After the initial reconstruction, the obtained model can be used to estimate concave regions in which the obtained photometric information is not accurate because of occlusions. Iteratively, the corresponding normals can be corrected and with improved normal maps the object can be reconstructed.

Bibliography

[1] J. Ackermann, F. Langguth, S. Fuhrmann, A. Kuijper, and M. Goesele. Multi-View Photometric Stereo by Example. In *International Conference on 3D Vision*, volume 1, pages 259–266, 2014.

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a Day. *Communications of the ACM*, 54(10):105–112, 2011.

[3] A. H. Ahmed and A. Farag. A New Formulation for Shape from Shading for Non-Lambertian Surfaces. In *Computer Vision and Pattern Recognition*, volume 2, pages 1817–1824, 2006.

[4] A. Ansar and K. Daniilidis. Linear Pose Estimation from Points or Lines. *Transactions on Pattern Analysis and Machine Intelligence*, 25(5):578–589, 2003.

[5] T. Bailey and H. Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[6] S. Bakshi and Y.-H. Yang. Shape from Shading for Non-Lambertian Surfaces. In *International Conference on Image Processing*, volume 2, pages 130–134, 1994.

[7] F. O. Bartell, E. L. Dereniak, and W. L. Wolfe. The Theory and measurement of bidirectional reflectance distribution function (BRDF) and bidirectional transmittance distribution function (BTDF). In *1980 Huntsville Technical Symposium*, pages 154–160, 1981.

[8] R. Basri and D. Jacobs. Photometric Stereo with General, Unknown Lighting. In *Computer Vision and Pattern Recognition*, pages 374–381, 2001.

[9] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric Stereo with General, Unknown Lighting. *International Journal of Computer Vision*, 72(3):239–257, 2007.

[10] B. G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, Stanford University, 1974.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[12] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[13] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille. The Bas-Relief Ambiguity. *International Journal of Computer Vision*, 35(1):33–44, 1999.

[14] M. Beljan, J. Ackermann, and M. Goesele. Consensus multi-view photometric stereo. In *DAGM/OAGM Symposium*, pages 287–296, 2012.

[15] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[16] P. J. Besl. Active, Optical Range Imaging Sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[17] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *Transactions on Pattern Analysis and Machine Intelligence*, pages 239–256, 1992.

[18] N. Birkbeck, D. Cobzas, P. Sturm, and M. Jägersand. Variational Shape and Reflectance Estimation under Changing Light and Viewpoints. In *European Conference on Computer Vision*, pages 536–549, 2006.

[19] J. R. Bitner, G. Ehrlich, and E. M. Reingold. Efficient Generation of the Binary Reflected Gray Code and Its Applications. *Communications of the ACM*, 19(9):517–521, 1976.

[20] J.-Y. Bouguet. Camera Calibration Toolbox for MATLAB, 2004. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, Last visited on Aug 8th, 2015.

[21] E. Boyer. Object Models From Contour Sequences. In *European Conference on Computer Vision*, pages 109–118, 1996.

[22] K. L. Boyer and A. C. Kak. Color-Encoded Structured Light for Rapid Active Ranging. *Transactions on Pattern Analysis and Machine Intelligence*, 9(1):14–28, 1987.

[23] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[24] D. C. Brown. Decentering Distortion of Lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.

[25] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[26] M. Byröd and K. Åström. Conjugate Gradient Bundle Adjustment. In *European Conference on Computer Vision*, pages 114–127, 2010.

[27] J. Canny. A Computational Approach to Edge Detection. *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[28] D. Caspi, N. Kiryati, and J. Shamir. Range Imaging With Adaptive Color Structured Light. *Transactions on Pattern Analysis and Machine Intelligence*, 20(5):470–480, 1998.

[29] M. K. Chandraker, S. Agarwal, and D. J. Kriegman. ShadowCuts: Photometric Stereo with Shadows. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[30] J. Y. Chang, K. M. Lee, and S. U. Lee. Multiview normal field integration using level set methods. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[31] Y.-C. Chang and J. F. Reid. RGB Calibration for Color Image Analysis in Machine Vision. *Image Processing*, 5(10):1414–1422, 1996.

[32] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Computer Vision and Pattern Recognition*, volume 1, pages 220–226, 2005.

[33] D. Coffin. Decoding raw digital photos in Linux. `https://www.cybercom.net/~dcoffin/dcraw/`, Last visited on Aug 8th, 2015.

[34] J. E. Cryer, P.-S. Tsai, and M. Shah. Integration of Shape from Shading and Stereo. *Pattern Recognition*, 28(7):1033–1043, 1995.

[35] W. B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized Voxel Coloring. In *Vision Algorithms: Theory and Practice*, pages 100–115, 2000.

[36] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH Computer Graphics*, pages 303–312, 1996.

[37] C. J. Davies and M. S. Nixon. Sensing Surface Discontinuities via Coloured Spots. In *International Workshop on Image and Signal Processing*, pages 573–576, 1996.

[38] J.-D. Durou, M. Falcone, and M. Sagona. Numerical Methods for Shape-from-Shading: A New Survey with Benchmarks. *Computer Vision and Image Understanding*, 109(1):22–43, 2008.

[39] H. Durrant-Whyte and T. Bailey. Simultaneous Localization and Mapping: Part I. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[40] S. C. Eisenstat. Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods. *SIAM Journal on Scientific and Statistical Computing*, 2(1):1–4, 1981.

[41] P. Eisert, E. Steinbach, and B. Girod. Multi-Hypothesis, Volumetric Reconstruction of 3-D Objects from Multiple Calibrated Camera Views. In *Acoustics, Speech, and Signal Processing*, volume 6, pages 3509–3512, 1999.

[42] C. Hernández Esteban and F. Schmitt. Silhouette and Stereo Fusion for 3D Object Modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[43] O. Faugeras and Q.-T. Luong. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, New edition, 2004.

[44] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *European Conference on Computer Vision*, volume 588, pages 563–578, 1992.

[45] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera Self-Calibration: Theory and Experiments. In *European Conference on Computer Vision*, volume 588, pages 321–334, 1992.

[46] P. Favaro and T. Papadhimitri. A Closed-Form Solution to Uncalibrated Photometric Stereo via Diffuse Maxima. In *Computer Vision and Pattern Recognition*, pages 821–828, 2012.

[47] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[48] A. Fitzgibbon and A. Zisserman. Automatic 3D Model Acquisition and Generation of New Images from Video Sequences. In *European Signal Processing Conference*, pages 311–326, 1998.

[49] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[50] R. T. Frankot and R. Chellappa. A Method for Enforcing Integrability in Shape from Shading Algorithms. *Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–451, 1988.

[51] Y. Furukawa and J. Ponce. Carved Visual Hulls for Image-Based Modeling. In *European Conference on Computer Vision*, pages 564–577, 2006.

[52] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multi-View Stereopsis. In *Computer Vision and Pattern Recognition*, 2007.

[53] Y. Furukawa and J. Ponce. Accurate Camera Calibration from Multi-View Stereo and Bundle Adjustment. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[54] H. Gärtner, P. Lehle, and H. J. Tiziani. New, highly efficient, binary codes for structured light methods. *Proc. SPIE*, 2599:4–13, 1996.

[55] M. Goesele, B. Curless, and S. M. Seitz. Multi-View Stereo Revisited. In *Computer Vision and Pattern Recognition*, volume 2, pages 2402–2409, 2006.

[56] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-View Stereo for Community Photo Collections. In *International Conference on Computer Vision*, pages 1–8, 2007.

[57] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, Third edition, 2006.

[58] M. Grochulla and T. Thormählen. Combining Photometric Normals and Multi-View Stereo for 3D Reconstruction. In *European Conference on Visual Media Production*, 2015.

[59] M. Grochulla, T. Thormählen, and H.-P. Seidel. Using Spatially Distributed Patterns for Multiple View Camera Calibration. In *International Conference on Computer Vision/Computer Graphics Collaboration Techniques*, pages 110–121, 2011.

[60] M. D. Grossberg and S. K. Nayar. Modeling the Space of Camera Response Functions. *Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1272–1282, 2004.

[61] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, volume 15, pages 147–151, 1988.

[62] R. I. Hartley. Estimation of Relative Camera Positions for Uncalibrated Cameras. In *European Conference on Computer Vision*, volume 588, pages 579–587, 1992.

[63] R. I. Hartley. Euclidean Reconstruction from Uncalibrated Views. In *Applications of Invariance in Computer Vision*, volume 825, pages 235–256, 1994.

[64] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Second edition, 2004.

[65] D. C. Hauagge, S. Wehrwein, K. Bala, and N. Snavely. Photometric Ambient Occlusion. In *Computer Vision and Pattern Recognition*, pages 2515–2522, 2013.

[66] G. E. Healey and R. Kondepudy. Radiometric ccd camera calibration and noise estimation. *Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, 1994.

[67] J. Heikkila and O. Silvén. A Four-step Camera Calibration Procedure with Implicit Image Correction. In *Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[68] C. Hernández, G. Vogiatzis, G. J. Brostow, B. Stenger, and R. Cipolla. Non-rigid Photometric Stereo with Colored Lights. In *International Conference on Computer Vision*, pages 1–8, 2007.

[69] C. Hernández, G. Vogiatzis, and R. Cipolla. Multiview Photometric Stereo. *Transactions on Pattern Analysis and Machine Intelligence*, pages 548–554, 2008.

[70] A. Hertzmann and S. M. Seitz. Example-Based Photometric Stereo: Shape Reconstruction with General, Varying BRDFs. *Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, 2005.

[71] O. Hesse. Die cubische Gleichung, von welcher die Lösung des Problems der Homographie von M. Chasles abhängt. *Journal für die reine und angewandte Mathematik*, 62:188–192, 1863.

[72] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *National Bureau of Standards*, 49(6), 1952.

[73] T. Higo, Y. Matsushita, N. Joshi, and K. Ikeuchi. A Hand-held Photometric Stereo Camera for 3-D Modeling. In *International Conference on Computer Vision*, pages 1234–1241, 2009.

[74] B. K. P. Horn. *Shape From Shading: A Method for Obtaining the Shape of a Smooth Opaque Object From One View*. PhD thesis, Massachusetts Institute of Technology, 1970.

[75] B. K. P. Horn. *Robot Vision*. MIT Press, 1986.

[76] B. K. P. Horn. Relative Orientation. *International Journal of Computer Vision*, 4(1):59–78, 1990.

[77] K. Ikeuchi. Determining Surface Orientations of Specular Surfaces by Using the Photometric Stereo Method. *Transactions on Pattern Analysis and Machine Intelligence*, 3(6):661–669, 1981.

[78] K. Ikeuchi and B. K. P. Horn. Numerical Shape from Shading and Occluding Boundaries. *Artificial intelligence*, 17(1):141–184, 1981.

[79] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A Radiosity Method for Non-Diffuse Environments. *SIGGRAPH Computer Graphics*, 20(4):133–142, 1986.

[80] International Telecommunication Union. *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, 2011. Recommendation BT.601-7 (03/11).

[81] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc., 1989.

[82] T. H. James. *Theory of the Photographic Process*. Macmillan, 1971.

[83] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A Practical Model for Subsurface Light Transport. In *Computer Graphics and Interactive Techniques*, pages 511–518, 2001.

[84] N. Joshi and D. J. Kriegman. Shape from Varying Illumination and Viewpoint. In *International Conference on Computer Vision*, pages 1–7, 2007.

[85] J. T. Kajiya. The Rendering Equation. *SIGGRAPH Computer Graphics*, 20(4):143–150, 1986.

[86] S. B. Kang, J. A. Webb, L. C. Zitnick, and T. Kanade. A Multibaseline Stereo System with Active Illumination and Real-time Image Acquisition. In *International Conference on Computer Vision*, pages 88–93, 1995.

[87] J. J. Koenderink and A. J. Van Doorn. The Generic Bilinear Calibration-Estimation Problem. *International Journal of Computer Vision*, 23(3):217–234, 1997.

[88] V. Kolmogorov and R. Zabih. Multi-camera Scene Reconstruction via Graph Cuts. In *European Conference on Computer Vision*, pages 82–96, 2002.

[89] V. Kolmogorov and R. Zabin. What Energy Functions Can Be Minimized via Graph Cuts? *Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[90] R. Kumar and A. R. Hanson. Robust Methods for Estimating Pose and a Sensitivity Analysis. *CVGIP: Image Understanding*, 60(3):313–342, 1994.

[91] K. N. Kutulakos and S. M. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[92] R. Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing Ltd, 2011.

[93] J. H. Lambert. *Photometria Sive de Mensura et Gradibus Luminus, Colorum et Umbrae*. Eberhard Klett, 1760.

[94] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[95] C. L. Lawson and R. J. Hanson. *Solving least squares problems*. Society for Industrial and Applied Mathematics (SIAM), 1987. Revised reprint.

[96] K. M. Lee and C.-C. J. Kuo. Shape from Shading with a Generalized Reflectance Map Model. *Computer Vision and Image Understanding*, 67(2):143–160, 1997.

[97] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, pages 164–168, 1944.

[98] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Computer Graphics and Interactive Techniques*, pages 131–144, 2000.

[99] M. Lhuillier and L. Quan. A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images. *Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433, 2005.

[100] J. Lim, J. Ho, M.-H. Yang, and D. Kriegman. Passive Photometric Stereo from Motion. In *International Conference on Computer Vision*, volume 2, pages 1635–1642, 2005.

[101] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[102] M. I. A. Lourakis and A. A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *Transactions on Mathematical Software (TOMS)*, 36(1):2nd, 2009. `http://www.ics.forth.gr/~lourakis/sba/`.

[103] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[104] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[105] B. D. Lucas and Takeo T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Artificial Intelligence*, pages 674–679, 1981.

[106] Q.-T. Luong and O. D. Faugeras. Self-Calibration of a Moving Camera from Point Correspondences and Fundamental Matrices. *International Journal of Computer Vision*, 22(3):261–289, 1997.

[107] W.-C. Ma, T. Hawkins, P. Peers, C.-F. Chabert, M. Weiss, and P. Debevec. Rapid Acquisition of Specular and Diffuse Normal Maps from Polarized Spherical Gradient Illumination. In *Eurographics Rendering Techniques*, pages 183–194, 2007.

[108] A. A. Magill. Variation in Distortion with Magnification. *Journal of the Optical Society of America*, 45(3):148–149, 1955.

[109] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[110] W. Matusik, C. Buehler, and L. McMillan. Polyhedral Visual Hulls for Real-Time Rendering. In *Workshop on Rendering*, pages 116–126, 2001.

[111] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-Based Visual Hulls. In *Computer graphics and interactive techniques*, pages 369–374, 2000.

[112] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. Image-Based 3D Photography Using Opacity Hulls. *Transactions on Graphics (TOG)*, 21(3):427–437, 2002.

[113] S. J. Maybank and O. D. Faugeras. A Theory of Self-Calibration of a Moving Camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[114] T. Mitsunaga and S. K. Nayar. Radiometric Self Calibration. In *Computer Vision and Pattern Recognition*, volume 1, 1999.

[115] J. Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. In *Numerical Analysis*, volume 630, pages 105–116, 1978.

[116] S. K. Nayar, K. Ikeuchi, and T. Kanade. Determining Shape and Reflectance of Hybrid Surfaces by Photometric Sampling. *Robotics and Automation*, 6(4):418–431, 1990.

[117] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently Combining Positions and Normals for Precise 3D Geometry. *Transactions on Graphics*, pages 536–543, 2005.

[118] F. E. Nicodemus. Directional Reflectance and Emissivity of an Opaque Surface. *Applied Optics*, 4(7):767–775, 1965.

[119] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. *Geometrical Considerations and Nomenclature for Reflectance*, volume 160. National Bureau of Standards, 1977.

[120] W. Niem and R. Buschmann. Automatic Modelling of 3D Natural Objects from Multiple Views. In *Image Processing for Broadcast and Video Production*, pages 181–193, 1995.

[121] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[122] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 2006.

[123] J. Park, S. N. Sinha, Y. Matsushita, Y.-W. Tai, and I. S. Kweon. Multiview Photometric Stereo using Planar Mesh Parameterization. In *International Conference on Computer Vision*, pages 1161–1168, 2013.

[124] S. Petitjean. A Computational Geometric Approach to Visual Hulls. *International Journal of Computational Geometry & Applications*, 8(4):407–436, 1998.

[125] M. Pollefeys, L. J. Van Gool, and A. Oosterlinck. The Modulus Constraint: A New Constraint for Self-Calibration. In *International Conference on Pattern Recognition*, pages 349–353, 1996.

[126] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbies, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[127] M. Pollefeys, R. Koch, and L. Van Gool. Self-Calibration and Metric Reconstruction Inspite of Varying and Unknown Intrinsic Camera Parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[128] J. L. Posdamer and M. D. Altschuler. Surface Measurement by Space-encoded Projected Beam Systems. *Computer Graphics and Image Processing*, 18(1):1–17, 1982.

[129] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Third edition, 2007.

[130] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, 1987.

[131] J. Salvi, S. Fernandez, T. Pribanic, and X. Llado. A state of the art in structured light patterns for surface profilometry. *Pattern Recognition*, 43(8):2666–2680, 2010.

[132] P. Saponaro, S. Sorensen, S. Rhein, A. R. Mahoney, and C. Kambhamettu. Reconstruction of Textureless Regions Using Structure from Motion and Image-based Interpolation. In *International Conference on Image Processing*, pages 1847–1851, 2014.

[133] K. Sato. Range Imaging System Utilizing Nematic Liquid Crystal Mask. In *International Conference on Computer Vision*, pages 657–661, 1987.

[134] K. Sato and S. Inokuchi. Three-dimensional Surface Measurement by Space Encoding Range Imaging. *Journal of Robotic Systems*, 2:27–39, 1985.

[135] T. Sattler, B. Leibe, and L. Kobbelt. SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter. In *International Conference on Computer Vision*, pages 2090–2097, 2009.

[136] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[137] D. Scharstein and R. Szeliski. High-Accuracy Stereo Depth Maps Using Structured Light. In *Computer Vision and Pattern Recognition*, volume 1, pages I:195–I:202, 2003.

[138] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006.

[139] S. M. Seitz and C. R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[140] J. R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994.

[141] W. M. Silver. Determining Shape and Reflectance Using Multiple Images. Master's thesis, Massachusetts Institute of Technology, 1980.

[142] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A Survey of Methods for Volumetric Scene Reconstruction from Photographs. In *Volume Graphics 2001*, pages 81–100, 2001.

[143] R. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[144] R. Smith, M. Self, and P. Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous robot vehicles*, pages 167–193, 1990.

[145] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *Transactions on Graphics*, volume 25, pages 835–846, 2006.

[146] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer Science & Business Media, 2013.

[147] P. Sturm and B. Triggs. A Factorization Based Algorithm for Multi-Image Projective Structure and Motion. In *European Conference on Computer Vision*, pages 709–720, 1996.

[148] R. Sturm. Das Problem der Projectivität und seine Anwendung auf die Flächen zweiten Grades. *Mathematische Annalen*, 1(4):533–574, 1869.

[149] S. Sullivan and J. Ponce. Automatic Model Construction and Pose Estimation From Photographs Using Triangular Splines. *Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1091–1097, 1998.

[150] R. Szeliski. Fast Shape from Shading. *CVGIP: Image Understanding*, 53(2):129–153, 1991.

[151] R. Szeliski. Rapid Octree Construction from Image Sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.

[152] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[153] R. Szeliski and S. B. Kang. Recovering 3D Shape and Motion from Image Streams using Non-Linear Least Squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

[154] H. D. Tagare and R. J. P. de Figueiredo. A Theory of Photometric Stereo for a Class of Diffuse Non-Lambertian Surfaces. *Transactions on Pattern Analysis and Machine Intelligence*, 13(2):133–152, 1991.

[155] A. J. Theuwissen. *Solid-State Imaging with Charge-Coupled Devices*. Kluwer Academic Press, 1995.

[156] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Technical report, Carnegie Mellon University, 1991.

[157] C. Tomasi and T. Kanade. Shape and Motion from Image Streams under Orthography: A Factorization Method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[158] P. H. S. Torr and D. W. Murray. The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix. *International Journal of Computer Vision*, 24(3):271–300, 1997.

[159] B. Triggs. Autocalibration and the Absolute Quadric. In *Computer Vision and Pattern Recognition*, pages 609–614, 1997.

[160] B. Triggs, P. McLauchlan, R. I. Hartley, and A. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In *Vision Algorithms: Theory and Practice*, volume 1883, pages 298–375, 2000.

[161] R. Y. Tsai. A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *Journal of Robotics and Automation*, 3(4):323–344, 1987.

[162] D. Vlasic, P. Peers, I. Baran, P. E. Debevec, J. Popovic, S. Rusinkiewicz, and W. Matusik. Dynamic Shape Capture using Multi-View Photometric Stereo. *Transactions on Graphics*, pages 1–11, 2009.

[163] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view Stereo via Volumetric Graph-cuts. In *Computer Vision and Pattern Recognition*, volume 2, pages 391–398, 2005.

[164] C. A. Wilson, A. Ghosh, P. Peers, J.-Y. Chiang, J. Busch, and P. Debevec. Temporal Upsampling of Performance Geometry Using Photometric Alignment. *Transactions on Graphics*, 29(2):17:1–17:11, 2010.

[165] L. B. Wolff, S. A. Shafer, and G. E. Healey. *Physics-Based Vision: Principles and Practice, Shape Recovery, Wellesley, MA, 1992*. A K Peter, Ltd/CRC Press, 1992.

[166] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19:139–144, 1980.

[167] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore Bundle Adjustment. In *Computer Vision and Pattern Recognition*, pages 3057–3064, 2011.

[168] C. Wu, Y. Liu, Q. Dai, and B. Wilburn. Fusing Multiview and Photometric Stereo for 3D Reconstruction under Uncalibrated Illumination. *Visualization and Computer Graphics*, 17(8):1082–1095, 2011.

[169] C. Wu, B. Wilburn, Y. Matsushita, and C. Theobalt. High-quality shape from multi-view stereo and shading under general illumination. In *Computer Vision and Pattern Recognition*, pages 969–976, 2011.

[170] C. Yang, J. Chen, N. Su, and G. Su. Improving 3D Face Details based on Normal Map of Hetero-source Images. In *Computer Vision and Pattern Recognition Workshops*, pages 9–14, 2014.

[171] L. Zhang, B. Curless, A. Hertzmann, and S. M. Seitz. Shape and Motion under Varying Illumination: Unifying Structure from Motion, Photometric Stereo, and Multi-view Stereo. In *International Conference on Computer Vision*, pages 618–625, 2003.

[172] L. Zhang, B. Curless, and S. M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. In *3D Data Processing Visualization and Transmission*, pages 24–36, 2002.

[173] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape-from-Shading: A Survey. *Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[174] Z. Zhang. A Flexible New Technique for Camera Calibration. *Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.