MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Measurement and Reconstruction of Visual Texture

Master's Thesis

KAREL CHLÁDEK

Brno, Spring 2022

MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Measurement and Reconstruction of Visual Texture

Master's Thesis

KAREL CHLÁDEK

Advisor: doc. RNDr. Pavel Matula, Ph.D.

Department of Visual Informatics

Brno, Spring 2022

FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Karel Chládek

Advisor: doc. RNDr. Pavel Matula, Ph.D.

Acknowledgements

First, I would like to thank my advisor, doc. RNDr. Pavel Matula, Ph.D., for numerous consultations and positive approaches throughout the thesis creation process. Second, I would like to thank my friend ing. Ladislav Bednář for the design of the circuit boards used in the thesis. Third, I want to thank my friends and family for their huge moral support. Last but not least, I want to thank the coffee shops and tea houses around my home for providing a nice and productive working environment.

Abstract

This work presents a method of visual texture measurement of photogrammetry reconstructions for usage in rendering and material analysis. The method creates uniformly sampled surface reflectance measurements of a mesh of real objects from images captured from multiple viewpoints under multiple light conditions. We design a cost-effective acquisition setup based on a light stage and assemble its prototype. We present a fast measurement estimation using the NVIDIA OptiX engine. We introduce an algorithm for measurement point sampling allowing a uniform measurement coverage of the measured object. This sampling also allows fast searching of reflectance of a general point on the surface of the mesh. We show an application of the obtained measurement by computing material reflectance models and introduce a method for directly rendering the material from a general viewpoint and general light condition.

Keywords

Photogrammetry, Visual Texture, BTF, BRDF, NVIDIA OptiX, Light stage

Contents

Introduction 1

1 Preliminaries 3
- 1.1 Visual Texture . . . 3
  - 1.1.1 General Reflectance Function . . . 3
  - 1.1.2 Bidirectional Texture Function . . . 4
  - 1.1.3 Bidirectional Reflectance Distribution Function . 5
  - 1.1.4 Simplyfied Reflectance Models . . . 5
- 1.2 3D Object Representation . . . 8
  - 1.2.1 Point Cloud . . . 8
  - 1.2.2 Mesh . . . 8
- 1.3 Photogrammetry . . . 10
  - 1.3.1 Pointcloud Reconstruction . . . 10
  - 1.3.2 Mesh Reconstruction . . . 11

2 Acquisition Setup 14
- 2.1 Acquisition Process . . . 14
- 2.2 Proposed Hardware . . . 15
  - 2.2.1 Light Stage . . . 16
  - 2.2.2 Object Manipulation . . . 23
  - 2.2.3 Setup Controlling . . . 24
- 2.3 Calibration . . . 24
  - 2.3.1 Camera Calibration . . . 24
  - 2.3.2 Light Calibration . . . 25

3 Proposed Method 27
- 3.1 Computation Pipeline . . . 27
- 3.2 Mesh Reconstruction and Viewpoint Estimation . . . 28
- 3.3 BTF Measurement . . . 28
- 3.4 Measurement Normalization . . . 30
- 3.5 Measurement Sampling . . . 31
  - 3.5.1 Barytex Forest Creation . . . 32
  - 3.5.2 Measurement Representative Searching . . . 35
  - 3.5.3 Valid Measurements Selection . . . 37
- 3.6 Material Model Computation . . . 38

3.7 Validation Rendering 39

4 Evaluation and Results **40**
4.1 Validation Dataset 40
4.2 Mesh Reconstruction and Measurement Validation . . . 42
4.2.1 Mesh Reconstruction Validation 42
4.2.2 Viewpoint Estimation and BTF Measurement Validation 44
4.3 Validation of Material Model Estimation 47

5 Conclusion **52**
5.1 Future Work 52

A Appendix **54**
A.1 Acquisition Setup Images 54

Bibliography **57**

List of Tables

4.1 Table of mean absolute errors for each object form the dataset. 47

List of Figures

1.1 Point cloud of the Stanford bunny. 9
1.2 Mesh of the Stanford bunny. 9
1.3 Mesh of a printed circuit board reconstructed by the ball pivoting algorithm (gray) from a given point cloud input (colored). Resulting mesh has many holes, but does not change the topology of the point cloud. 12
1.4 Mesh of a printed circuit board reconstructed by the Screened Poisson algorithm (gray) from a given point cloud (colored). Resulting mesh has no holes but creates an artificial brim around the object. 13

2.1 Once subdivided truncated icosahedron. 17
2.2 Model of the acquisition setup. 18
2.3 Light module with and without a led board. 18
2.4 Camera module. 19
2.5 Camera module with camera attached. 19
2.6 Detail of a module connector. 20
2.7 Usage of a module connector. 20
2.8 Assembled acquisition setup. 21
2.9 Access to the inside of the light stage. 22
2.10 Placing of light boards inside the light stage. The unused section of modules is overlaid by red color. 22
2.11 Turn table inside the light stage. 23

3.1 Comparison of the original reconstructed mesh (top) with the used part of the mesh (bottom). 29
3.2 Material tree illustrations: triangle subdivision grid (top image), tree structure (middle image, each node represents a triangle), data alignment (bottom) 32
3.3 Ordering of the measurement points (left) and the 2D index mapping (right). 33
3.4 Visualisation of the measurement tree search sectors. Notation is as follows: current triangle ABC, divisor points D_a, D_b, D_c. 36

3.5 Visualisation of the error of Euclidean metric when choosing the best measurement for a sample point (red). The measurement on the blue point is the closest in space, but the measurement in the green point is better choice. . . . 38
4.1 Validation dataset: PCB (top left), rock (top right), coin (bottom left) and box (bottom right). 41
4.2 Example of input images for the rock object. 41
4.3 Visualization of the reconstructed mesh of the rock object. 42
4.4 Visualization of the reconstructed mesh of the PCB object. 43
4.5 Visualization of the reconstructed mesh of the coin object. 43
4.6 Visualization of the reconstructed mesh of the box object. 44
4.7 Images of the coin object, captured from multiple viewpoints, overlaid with a red render of the reconstructed mesh. 45
4.8 Measurements obtained from a single image of the rock object in the form of a colored point cloud. 46
4.9 Comparison of real images to the rendered images (left) using estimated Lambert reflectance model (right). 49
4.10 Comparison of real images to the rendered images (left) using estimated Blin-Phong reflectance model (right). . . 50
4.11 Comparison of real images to the rendered images (left) using estimated Cook-Torrance reflectance model (right). 51
A.1 Acquisition setup with active blue global illumination. . . 55
A.2 Inside view of the acquisition setup with active blue global illumination. 55
A.3 Model of the acquisition setup from the side view. 56
A.4 Model of the acquisition setup from the top view. 56
A.5 Model of the acquisition setup from the inside. 57

Introduction

One of the biggest challenges of computer vision and computer graphics is reconstructing the high quality digital representation of real 3D objects from one or more images. The digital reconstruction can be divided into two main parts: 3D shape and object's material properties (texture).

In the past, multiple methods of shape reconstruction were developed based on laser or photogrammetry scanning. These methods often lack good surface texture details, mainly surface reflectance (i.e., color, roughness, subsurface scattering, and transparency). Such estimation can be used for tasks like 6D pose estimation, robotic navigation, or structural analysis. On the other hand, there is little to no relevant information provided by these types of 3D scanning for tasks like material analysis and rendering from a general viewpoint under general light conditions.

Laser scanning provides no material information, as most of these scanners lack any visual observation and computes the point cloud of an object by casting light rays in the scene.

The photogrammetry is a technique of reconstruction based on multiple images captured from unique viewpoints. This method represents the reconstructed real object in the form of a colored point cloud, which, as a material property, is sufficient only for global ambient lighting and in cases where the point cloud is dense enough to represent all surface and texture details.

Currently the most often used devices of material and texture acquisition are gonioreflectometers [1] and light stages [2]. These methods are thoroughly described by Haindel et al. [3]. It is possible to acquire precise measurements of object reflectance using these setups. However, it can be challenging to map these measurements precisely on the surface of an object with complex geometry. Also, building these setups can get quite costly as it requires either multiple precise moving parts or multiple cameras.

Therefore this thesis combines photogrammetry, as the most affordable way of 3D reconstruction from unknown viewpoints, with a light stage, as a setup without camera movement relative to light sources. The resulting setup, or its versions, can then be statically

mounted on a robotic arm (for automated industrial purposes), or handheld (for manual usage). The proposed measurement method maps each observation pixel as a separate reflectance observation onto the reconstructed mesh. The mesh is than uniformly sampled and each sample is stored with its valid observations into a newly introduced data structure. Finally, there is shown a method of computing conventional reflectance models from reflectance measurements, as dense sampled measurement is memory demanding and computationally heavy for usage in real-time rendering.

The thesis is structured as follows. In chapter 1, we lay down the theoretical basis and terminology. In chapter 2, there is described an acquisition setup design and its calibration. In chapter 3 we introduce the method for material reflectance measurement, and in chapter 4 we validate the results by comparing synthetic reconstructions with real images. Chapter 5 then discuss the results, further work, and possible improvements of the proposed method.

1 Preliminaries

This chapter describes a theoretical background further used in the proposed measurement method. First, a visual texture is defined as the main topic of this thesis. Then there is a short introduction to 3D shape representations to define proper notation. Finally, photogrammetry is described with its usage for the reconstruction of a point cloud, camera calibration, and image view matrices.

1.1 Visual Texture

Visual texture is a material property that describes a behavior of a light interacting with the material, thus making up a big part of the material's visual appearance. When the texture is measured precisely, it can provide information about an object's behavior in different illumination conditions and information needed to analyze or recognize observed materials. In the real world, the visual texture can be simplified to a light behavior on the border of two isotropic optically different materials because the reflection of a light ray hits microscopic flat isotropic parts of the observed surface (microsurfaces). Computer representation of a real object is not usually that dense to incorporate all microsurfaces and volumetric properties of the object. Therefore the texture often incorporates properties caused by observing an area containing many microsurfaces, such as diffuse reflectance (microsurfaces are randomly oriented) or anisotropy (microsurfaces are oriented more often in a particular direction).

This section describes the generic reflectance function as a mathematical representation of a visual texture. The generic reflectance function is too computationally complex for most texture analysis tasks, so further in this section, there are typical simplifications and its data representation. Definitions and notations are inspired by Haindel et al. [3].

1.1.1 General Reflectance Function

The general reflectance function (GRF) is a 16-dimensional function that describes the percentage of intensity of a reflected (or transmitted)

light from an illuminated surface. It is currently too complex to use, so it serves as a theoretical background for further simplifications based on the assumptions about observed material.

The GRF can be written as a function

$$Y^{GRF} = GRF(\lambda_i, x_i, y_i, z_i, t_i, \theta_i, \phi_i, \lambda_v, x_v, z_v, t_v, \theta_v, \phi_v, \theta_t, \phi_t), \quad (1.1)$$

where λ_i is a wavelength of the light ray incomming from the direction $[\theta_i, \phi_i]$ (elevation θ and azimuth ϕ)in time t_i. Coordinates of the illuminated surface are $[x_i, y_i, z_i]$. Outgoing light of wavelength λ_v is observed in direction $[\theta_v, \phi_v]$ on surface at coordinates $[x_v, y_v, z_v]$ with transmision direction θ_t, ϕ_t.

For convinience there is further used $\mathbf{X} = [x, y, z]$ to describe coordinates and $\omega = [\theta, \phi]$ to describe direction as an incidence angle. With this notation the GRF can be rewritten as

$$Y^{GRF} = GRF(\lambda_i, \mathbf{X}_i, t_i, \omega_i, \lambda_v, \mathbf{X}_v, t_v, \omega_v, \omega_t). \quad (1.2)$$

1.1.2 Bidirectional Texture Function

Bidirectional texture function (BTF)is a state-of-the-art 7-dimensional approximation of GRF. It is often used because it can be simultaneously measured and modeled. For this property, this function is further used in this thesis.

This function uses further assumptions:

1. The observed surface is the same as the illuminated surface ($\mathbf{X}_i = \mathbf{X}_v = \mathbf{X}$).
2. The reflectance is observed in a static timeframe ($t_i = t_v = t$).
3. The reflectance of the surface is not changing in time ($t = 0$).
4. The reflectance does not change the wavelength ($\lambda_i = \lambda_v = \lambda$).
5. No light is transmitted through the surface (ω_t is not used).
6. Light intensity does not change by the distance from the source ($X = [x, y]$).

With these assumptions the function is written as:

$$Y^{BTF} = BTF(\lambda, \mathbf{X}, \omega_i, \omega_v), \tag{1.3}$$

with same notation as Eq. 1.2.

1.1.3 Bidirectional Reflectance Distribution Function

Bidirectional reflectance distribution function (BRDF)is a further approximation of the BTF, which assumes that the reflectance is invariant on the position of the observed surface. As such it is ideal for modeling the reflectance of an uniform object or a single surface. The BRDF is than written as 5-dimensional function

$$Y^{BRDF} = BRDF(\lambda, \omega_i, \omega_v), \tag{1.4}$$

with the same notation as Eq. 1.3.

To lower dimensionality of the BRDF it can be assumed that for each measured wavelength there is a different function

$$Y^{BRDF}_{\lambda} = BRDF_{\lambda}(\omega_i, \omega_v). \tag{1.5}$$

As most cameras use only the RGB spectrum to manage colors the BRDF is often divided into three separete functions

$$\begin{aligned} Y^{BRDF}_{R} &= BRDF_{R}(\omega_i, \omega_v), \\ Y^{BRDF}_{G} &= BRDF_{G}(\omega_i, \omega_v), \\ Y^{BRDF}_{B} &= BRDF_{B}(\omega_i, \omega_v), \end{aligned} \tag{1.6}$$

lowering the dimensionality of BRDF from the 5-dimensional into a 4-dimensional.

1.1.4 Simplyfied Reflectance Models

The measured BRDF can be too computationally expensive for many tasks (e.g. rendering or photometric stereo). Many reflectance models that encode texture properties into a few parameters have been developed in the past. In this chapter, there are shown some of the typically used reflectance models.

Lambertian Reflectance Model

Lambertian reflectance model (LRM), often referred to as ideally diffuse reflectance model, is the simplest reflectance model, which assumes that the surface reflects light uniformly in all directions. The only parameter that describes a material in this model is called albedo. Albedo compresses BRDF into a single value that represents an inverse of light absorbance of the observed surface.

Light reflectance is than a function:

$$Y_{\lambda}^{LRM}(L) = \rho_{\lambda}(N \cdot L), \tag{1.7}$$

where ρ is the albedo, N is the unit surface normal vector and L is the unit vector representing direction to a light source. Note that LRM is not dependent on the observing direction and L, as unit vector, can be written in form of ω so the resulting function is 2-dimensional.

This model is often used in inverse rendering methods (normal estimation) such as photometric stereo for its simplicity but introduces a significant error because real materials contain specular reflectances. LRM is also physically incorrect as the sum of incoming light energy can be significantly lower than the sum of the reflected light energy.

Blin-Phong Reflectance Model

Blin-Phong reflectance model (BPRM) is an extension of LRM that incorporates a basic specular reflectance and dependency on the viewing direction.

To incorporate viewing direction it uses a half vector

$$H(L, E) = \frac{L + E}{||L + E||}, \tag{1.8}$$

where L is a vector in the direction from surface to a light source and E is a vector from surface into an eye (viewing direction).

The Blin-Phong reflectance model is a function

$$Y_{\lambda}^{BPRM}(L, E) = Y_{\lambda}^{LRM}(L) + K_{s,\lambda} \times (N \cdot H(L, E))^{\alpha_{\lambda}}, \tag{1.9}$$

where $K_{s,\lambda}$ and α are coefficients of material specularity and shininess. The specularity influences the intensity of the specular reflection, and the shiningness influences the size of a specular lobe.

Overall this model encodes the material reflectance into three parameters (albedo, specularity, shiningness). It has been used in many applications requiring fast computation and more realistic light reflections than LRM, but it is insufficient for modern or physically based rendering.

Cook–Torrance Reflectance Model

Cook–Torrance reflectance model [4] (CTRM) is a physically based reflectance model that uses microfacets to represent the observed area. It is used to approximate a specular reflectation.

This model is composed of three main parts: Distribution therm, Fresnel term and Geometry therm.

As the Distribution term is often used Beckmann distribution [5]

$$D_{Beckmann}(m,\alpha) = \frac{exp(-tan^2(\alpha)/m^2)}{\pi m^2 cos^4(\alpha)}, \tag{1.10}$$

where m is roughness of the material (slope of the microsurfaces) and

$$\alpha = arccos(N \cdot H), \tag{1.11}$$

is an angle between a unit normal vector N and a half vector H (Eq. 1.8).

The Fresnel term causes that light rays close to parallel to the observed surface have stronger specular reflections. In most of the applications this term is approximated by Schlick's approximation [6]

$$F_{Schlick}(F_0, L, N) = F_0 + (1 - F_0)(1 - L \cdot N), \tag{1.12}$$

where L is an unit vector in direction of outgoing light, N is an unit normal vector and F_0 is a Fresnell reflection for angle $0°$ between L and N.

The Geometry term is a function

$$G(N,E,L,H) = min\left\{1, \frac{2(N \cdot H)(N \cdot E)}{E \cdot H}, \frac{2(N \cdot H)(N \cdot L)}{E \cdot H}\right\}. \tag{1.13}$$

The overall Cook-Torrance specular reflections is

$$Y_\lambda^{CTRM}(N,E,L) = \frac{D(m,\alpha)F(F_0,L,N)G(N,E,L,H)}{4(N \cdot L)}. \tag{1.14}$$

CTRM does not represent a diffuse reflections so it is often combined with other models such as LRM:

$$Y_{lambda}(N, E, L) = Y_{\lambda}^{LRM}(N, L) + Y_{\lambda}^{CTRM}(N, E, L). \tag{1.15}$$

1.2 3D Object Representation

3D objects can be digitally interpreted in multiple ways. This section describes the most typical representations of 3D objects: point cloud and mesh. For purposes of this thesis, we discard analytical representations. They are often used for CAD modeling but are not suitable for photogrammetry reconstructions.

1.2.1 Point Cloud

The point cloud is the most straightforward object representation. It consists of a set of points on the surface of the represented object. Each point consists of its coordinates and optionally additional data such as surface normal and the color of the point. Point clouds are the most often used object representation for 3D scene reconstruction, as most scanners or methods discretely find points on the surface. An example of a point cloud can be seen in Fig. 1.1.,

1.2.2 Mesh

Mesh is the most widely used representation of 3D objects. In addition to points, it also represents the surface of the object. Therefore a mesh is more suitable for texturing than a point cloud.

Mesh consists of three sets: Vertices, Edges, and Faces. Vertices are points on the surface of the object. They are connected by edges. Faces are created by connecting three or more vertices by edges. The most often used mesh variant is a triangular mesh that has to create planar faces as three vertices describe a plane. All faces can be triangulated, so for the purposes of this thesis, a mesh is considered to be triangular.

Example of the mesh can be found in Fig. 1.2.

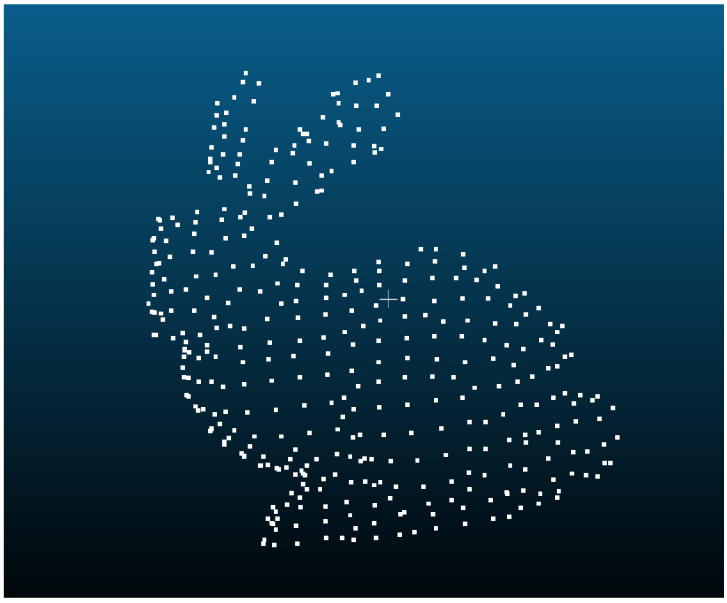

Figure 1.1: Point cloud of the Stanford bunny.

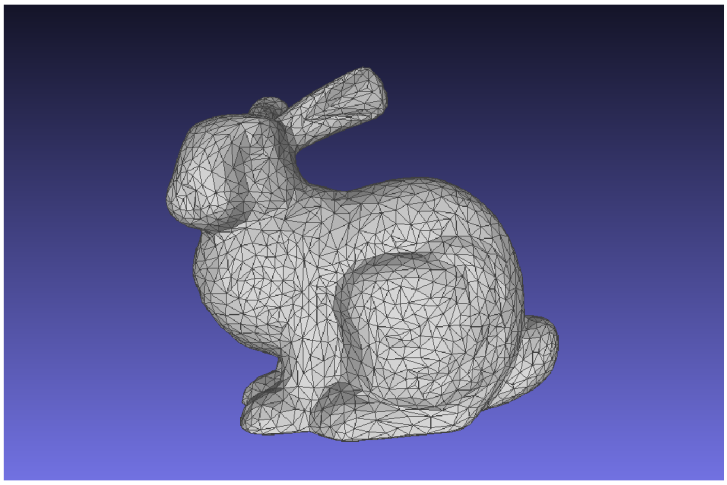

Figure 1.2: Mesh of the Stanford bunny.

1.3 Photogrammetry

Photogrammetry is a technique for obtaining reliable information about a real scene. For purposes of this thesis, the term photogrammetry is used to describe a specialization called Stereophotogrammetry. Stereophotogrammetry is a technique of reconstructing a point cloud of a real scene from multiple images captured from multiple viewpoints (this technique is also often referred to as Structure-from-Motion). This section briefly describes the basics of photogrammetry and mesh reconstruction from a point cloud.

1.3.1 Pointcloud Reconstruction

Pointcloud in photogrammetry can be reconstructed in numerous ways. This thesis uses software called COLMAP which uses methods introduced by Schoenberger et al. [7, 8, 9], the further described computational pipeline is motivated by these methods.

The first step to reconstructing a point cloud using photogrammetry is locating important pixels in captured images. Each pixel represents a projection of a 3D point into the image plane. The importance of the pixel can be assigned in multiple ways. The structurally important points (corners and edges) or texturally important points (changes in texture) are often detected. Important pixels are often detected by feature detectors such as SIFT, SURF, ORB, or, more recently, by neural networks.

The second step is finding corresponding pixels (matches) between images and computing the base point cloud. Finding matches can be done with numerous methods. For example, it can be done using RANSAC, matching algorithms based on global optimizations (often initialized by RANSAC), or neural networks. The 3D point position can be computed from two corresponding pixels using epipolar geometry, intrinsic camera matrix, and view matrix (extrinsic camera matrix). The view matrix is often unknown in real applications, so the transformation from the first position of the camera into the next one is computed first. It can be done by aggregating multiple matches between two images and solving a system of linear equations (this process is called Bundle adjustment). To further improve the estimation of the view matrix, the system can be extended by multiple views

and matches across all (or valid selection) of the images. For cases where the intrinsic camera matrix is unknown, it can be solved inside this process, but more matches are required.

Point cloud from the second step has many outliers caused mainly by wrong matching or imprecise detection of important points, so the third step is often outlier removal. An outlier can be found by observing distance and geometric correspondences between point estimation from different pairs of images and geometric relations between close points. After outlier removal, it can be iterated between steps two and three if needed.

After step three, the point cloud is usually really sparse. Depending on the application, the fourth step can be densification and improving the resulting point cloud. In the case of COLMAP, the densification is done by a hierarchical ordering of the scene and by using PatchMatch [10] to find more and better matches.

1.3.2 Mesh Reconstruction

For proper mapping of the texture on the whole surface of the object, it is needed to create a mesh from the reconstructed point cloud. Currently, the most often used meshing algorithms are Ball pivoting and Screened Poison surface reconstruction.

Ball Pivoting

Ball pivoting [11] is a meshing algorithm based on the idea of a ball of a user given radius pivoting around an edge of a triangle. If the ball touches three points and does not contain any other point, these three points represent a face. The algorithm starts with a choice of one triangle and one edge. The ball pivots around the selected edge until it touches a point, creating a new face. All edges (except already searched ones) are added to a queue. The next pivoting edge is chosen, and the process repeats until no edge is left. If there are multiple groups of points in a distance larger than the diameter of the pivoting ball, the algorithm repeats for each group.

This algorithm is often used in photogrammetry, especially when large scenes are reconstructed, as it preserves holes between reconstructed objects. It also has a small memory requirement and a simple

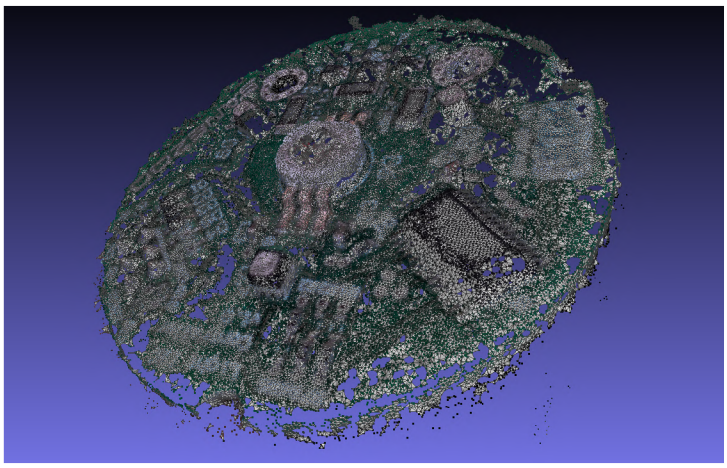

Figure 1.3: Mesh of a printed circuit board reconstructed by the ball pivoting algorithm (gray) from a given point cloud input (colored). Resulting mesh has many holes, but does not change the topology of the point cloud.

implementation. The biggest problems of this algorithm are sensitivity to the point cloud imperfections and the strong dependency on the choice of the radius of the ball (see Fig. 1.3).

Screened Poison Surface Reconstruction

Screened Poison surface reconstruction (SPSR) [12] is a meshing algorithm based on a global optimization. It is an extension of the Poisson surface reconstruction (PSR) [13].

Poisson surface reconstruction uses points, with normals, from a given point cloud to construct a vector field $\overrightarrow{V}$. Than the algorithm finds a scalar function $\mathbf{u}$ that minimizes energy:

$$E(\mathbf{u}) = \int ||\nabla \mathbf{u}(p) - \overrightarrow{V}(p)||^2 \, dp, \tag{1.16}$$

where p denotes a position. In practice $\mathbf{u}$ is often chosen to be a linear combination of multiple B-splines. Therefore for basis $\{B_1, ..., B_N\}$ with weights $\{x_1, ..., x_N\}$ the scalar function $\mathbf{u}$ can be simplified into

$$\mathbf{u} = \sum_{i=1}^{N} x_i B_i. \tag{1.17}$$

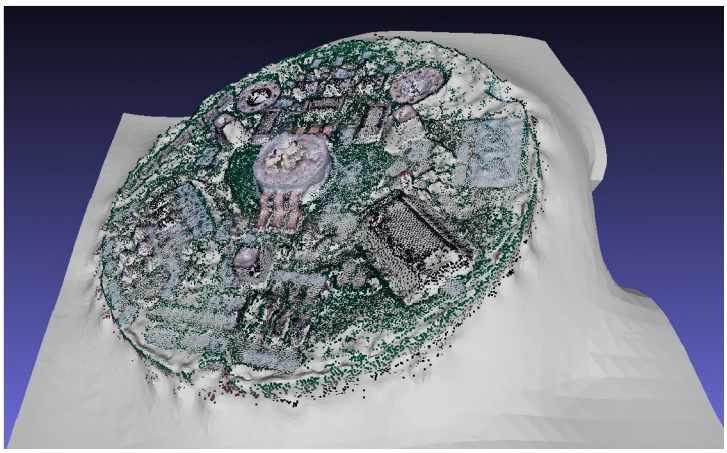

Figure 1.4: Mesh of a printed circuit board reconstructed by the Screened Poisson algorithm (gray) from a given point cloud (colored). Resulting mesh has no holes but creates an artificial brim around the object.

Parameters of the B-splines and their weights can be optimized using the Euler-Lagrange equation.

After the $\mathbf{u}$ is computed, the implicitly described function can be sampled by a grid of points. These samples are than converted into a mesh using the grid.

SPSR extends the minimized energy by a term for closeness to an initialization plane (screen). This term gives the optimization better stability and time efficiency, but can leave some artificial fragments of the initialization plane in the resulting mesh (see Fig. 1.4).

Both SPSR and PSR are used mainly because they create a water-tight surface and they can filter high frequency errors in the given point cloud. This algorithm is used in this thesis because it fills holes in the reconstructed point clouds better than the Ball pivoting algorithm.

2 Acquisition Setup

Visual texture can be acquired in multiple ways. The most often used setups are light stages and gonioreflectometers.

The light stage is a static dome-like structure with multiple light sources and cameras placed on the inner surface of the structure. The observed object is then placed inside the dome and captured with each camera and light source combination. For time efficiency, multiple cameras and multiple lights are often triggered at the same time. Gonioreflectometer has only one light source and one camera placed on two separate moving arms around a capturing table. The observed object is placed on the capturing table. The light source and camera are moved to multiple locations on a hemisphere around the capture table and for each position acquires an image. The size of the hemisphere is often static and given by the length of the moving arms.

Both setups at the end produce an array of images with different illumination for each unique camera position. Each image contains information about the position of the light source and the camera in the scene, which is precisely the data needed for the visual texture reconstruction.

This chapter aims to design a generic, cost-effective acquisition setup prototype that will produce comparable results to the setups mentioned above and allow the development of a specialized setup depending on the application.

At first, the acquisition process is described with its theoretical reasoning. Further, the proposed setup's construction and hardware design are described. Finally, there are described methods to calibrate the setup to produce valid information about positions in the scene.

2.1 Acquisition Process

For proper measurement of BTF, it is needed to measure BRDF on as many points on the observed object. For measuring BRDF, it is needed to sample the space of parameters of BRDF as densely as possible to allow the fitting of complex reflectance models. Therefore the goal is to capture the object from as many viewpoints as possible and to have as many unique light sources for each viewpoint as possible. Views

should be overlapping to allow proper photogrammetry reconstruction and increase the number of unique observing directions for each point on the surface of the observed object.

Therefore the requirements for the setup are:

1. Each viewpoint has a dense coverage of light sources around the captured object.

2. The object is captured from many overlapping viewpoints.

3. Captured images should cover a significant part of the object's surface (ideally the whole surface).

The first requirement can be satisfied by adding multiple moving parts or mounting the light sources statically into the scene or the camera itself. In this setup, it has been chosen to mount the camera and light sources on the same construction. For the second and third requirements, adding multiple cameras into the scene or moving with either camera or the scene is needed. Using multiple cameras can get quite costly, and it does not allow for an increase in the viewpoint coverage density if needed. Therefore it has been chosen to add movement.

For purposes of this thesis, it has been chosen to use a static light stage with a single camera as it covers the first requirement with the lowest price. The camera's movement is then simulated by moving the observed object as the movement of a larger light stage is too difficult to manage. The disadvantage of this approach is that it does not work correctly for nonrigid objects as they often change their shape with movement. For the whole acquisition process, see the pseudocode (Algorithm 1).

2.2 Proposed Hardware

This section describes the hardware design and assembly of the acquisition setup. For clarity, the setup is divided into two main parts: light stage and object manipulation.

Algorithm 1 Acquisition process

```
for each object position do
    Move object to the position.
    Turn on global light.
    Capture photogrammetry reconstruction image.
    Turn off global light.
    for each light source do
        Turn on the light source.
        Capture an image.
        Turn off the light source.
    end for
end for
```

2.2.1 Light Stage

The light stage, as described above, is a spherical, or hemispherical, construction with light sources and cameras (in this case, one camera) placed on its surface. Further are described main parts and their assembly.

Construction

Construction of a light stage is most often done by creating a geodesic dome that uniformly covers a sphere either by the centers of its faces or by its corner points.

A geodesic dome is a shell structure in the shape of a geodesic polyhedron. A geodesic polyhedron is created by an iterative subdivision of an icosahedron and projection of the new vertices into a sphere. Therefore the vertices of a classical geodesic dome lay on the surface of a sphere. By this subdivision, a triangular grid approximating a sphere is created. It is challenging to place cameras and lights into corners of a structure, so a subdivided truncated icosahedron is used, covering the sphere with pentagons and hexagons. Concretely there is used once subdivided version, which covers the surface by four different polygons: equilateral pentagon, equilateral hexagon and two different generic hexagons (see Fig. 2.1).

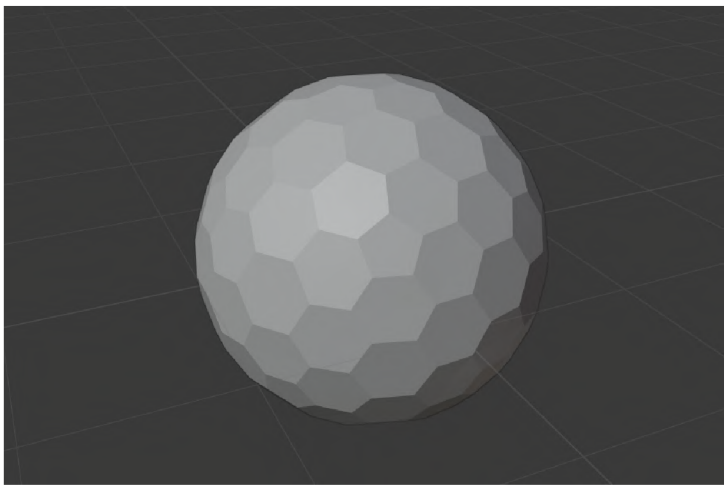

Figure 2.1: Once subdivided truncated icosahedron.

The most often used construction of a geodesic dome is based on struts and hubs. This construction first creates a frame where hubs are placed into the vertices of the dome. Hubs are connected by struts, thus creating the wireframe of the dome. The second step, if needed, is to fill the faces with flat covers. This construction requires a lot of different parts and adjustments to create a precise geodesic dome.

Therefore the proposed construction is face-based. Dome is divided into modules (one module for each face). We need four different module shapes for each different face shape for this construction. Modules are a thicker version of a face where sides are slightly tilted to fit precisely into the neighbor modules. By doing this, the shape of the dome is preserved by pressure between the modules. There are two specializations of modules: camera module and light module. The light module (see Fig. 2.3) has a mount for a light source, a cut out with the shape of the light source's printed circuit board (PCB), and a space for air cooling. Camera module (see Figures 2.4 and 2.5) has a hole for camera lens and camera mount. Full model of the light stage can be found in Fig. 2.2 (for different views see Appendix A.1). To hold the modules together there are connectors (see Figures 2.6 and 2.7) at each vertex of the dome.

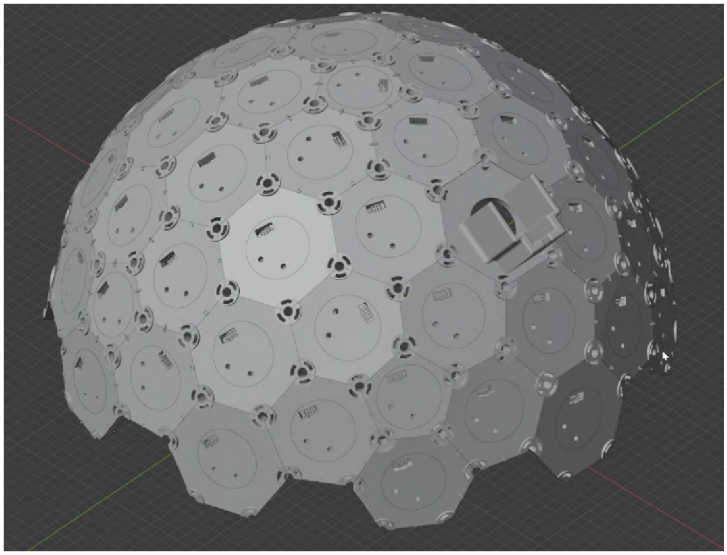

Figure 2.2: Model of the acquisition setup.

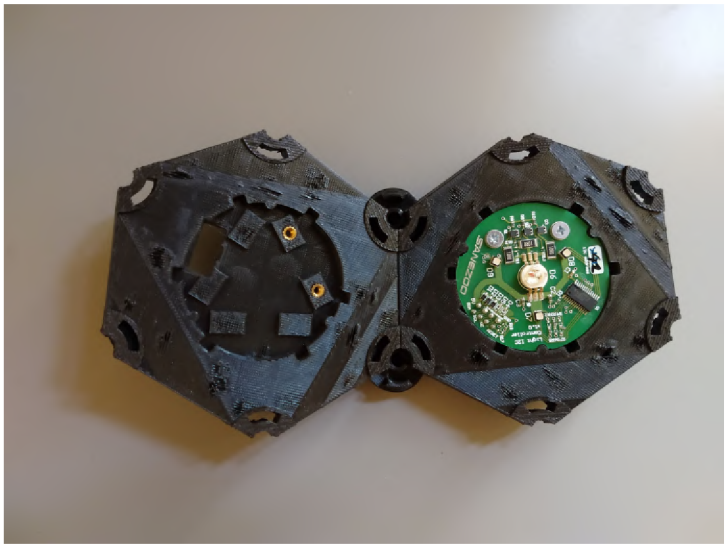

Figure 2.3: Light module with and without a led board.

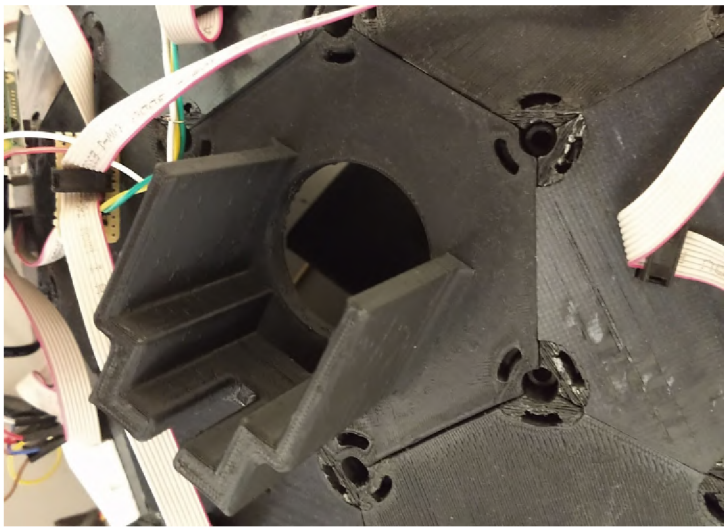

Figure 2.4: Camera module.

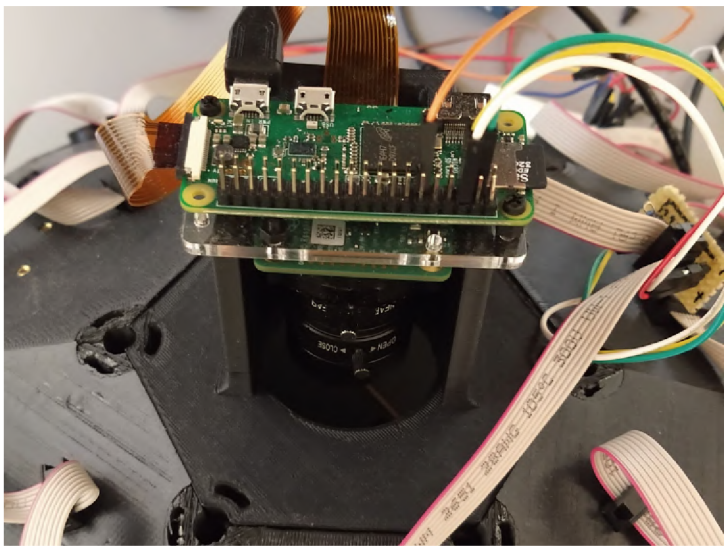

Figure 2.5: Camera module with camera attached.

Figure 2.6: Detail of a module connector.

Figure 2.7: Usage of a module connector.

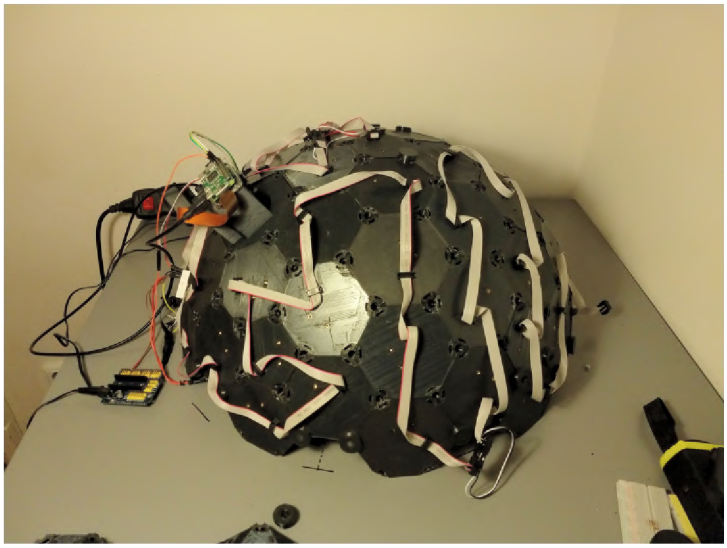

Figure 2.8: Assembled acquisition setup.

Light modules create the whole hemispherical construction except for one camera module (see Fig. 2.2). There is a removable plate to better manipulate an object inside the dome (see Fig. 2.9).

This construction can be fully 3D printed. It is as precise as the used 3D printer, in this case Creality Ender 3 V2. Printed and assembled is shown in the Fig. 2.8 (for images with active light sources see Appendix A.1).

Light Sources

There were custom created I2C LED light modules for light sources with three low-power LEDs and one high-power one. Each led is RGB and can be addressed and dimmed separately from others. Board can be seen in Fig. 2.3 on the right.

Due to multiple manufacturing errors, there were not enough functioning boards to cover the whole dome (see Fig. 2.10 for placing of functioning boards). Therefore the count of actively used boards was lowered from possible 55 to 37 (148 possible unique RGB light sources).

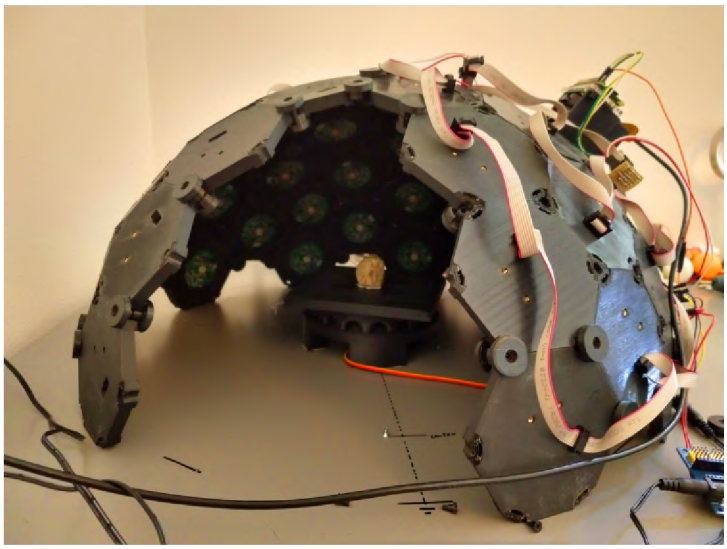

Figure 2.9: Access to the inside of the light stage.

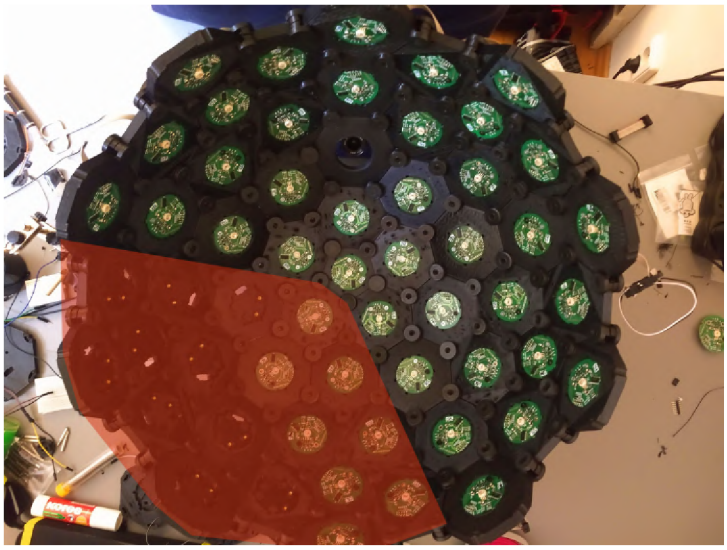

Figure 2.10: Placing of light boards inside the light stage. The unused section of modules is overlaid by red color.

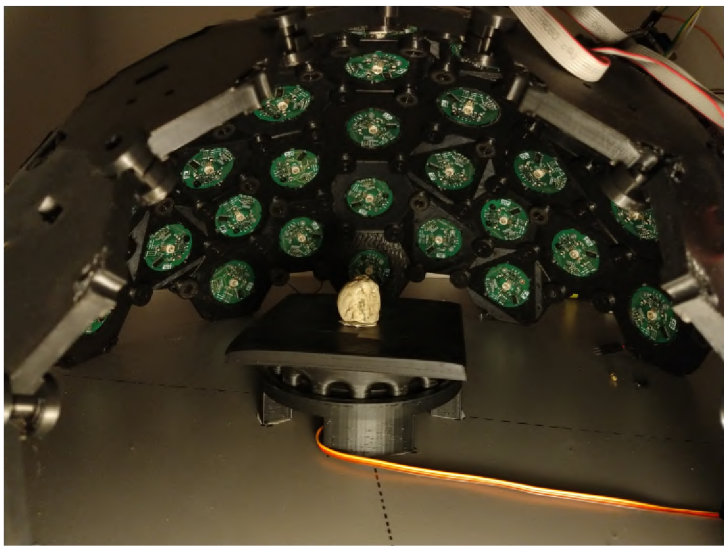

Figure 2.11: Turn table inside the light stage.

Camera

It is preferable to use a high-resolution camera, which extends the reconstructed texture's resolution. This setup uses Raspberry Pi RASP CAM HQ (see Fig. 2.5) as it is an affordable high resolution camera using c-mount. It also has a good acquisition library for a simple usage.

2.2.2 Object Manipulation

To extend the number of views, it is needed to move the captured object, ideally as automatically as possible. This setup uses a turntable as simple, but for prototyping, a sufficient manipulator. The turntable is assembled from three parts: tabletop, pedestal, and servo. Servo mounted in the pedestal rotates the tabletop resting on the pedestal (see Fig. 2.11). To improve the precision and speed of the turn table, there can be used ball bearings between the tabletop and the pedestal.

The choice of a turntable comes with some limitations. The first and most significant limitation is the inability to rotate the object around different than the z-axis, requiring (if needed) manual rotation of the object on the tabletop and rerunning the capturing process. The

second limitation is that the tabletop itself often shadows parts of the object when illuminated from under the table.

2.2.3 Setup Controlling

The setup is controlled by a Raspberry PI Zero W. There are three separate power sources (the camera with raspberry, light sources, and the turntable). Light modules are all connected using an I2C bus to the Raspberry PI. Servo from the turntable is connected to the Raspberry PI via GPIO pin. The camera is connected using a flex cable. The Raspberry PI is then remotely controlled through the wifi.

2.3 Calibration

The proposed method setup needs to be calibrated to provide valid information about the acquisition conditions. In this section, camera calibration and light calibration are discussed.

2.3.1 Camera Calibration

For precise mapping of the BRDF measurements onto the surface of the observed object, the camera matrix (or its inverse) is needed. A camera matrix is a projection matrix representing a mapping from the real world into an image plane.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = K \cdot V \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2.1}$$

where $[x, y, z]$ are real point coordinates, $[u, v]$ are coordinates of the points projection into the image plane, M is a camera matrix, K is an intrinsic camera matrix, representing the actual projection, and V is a view matrix (extrinsic matrix), representing a transformation from the world coordinates into the camera coordinates.

The view matrix is optimized using the photogrammetric software because the software usually constructs its world coordinate system for each reconstruction.

The intrinsic camera matrix can be obtained in two ways. It can be optimized outside of the setup using images of a calibration pattern such as a chessboard or a ChArUco board. In the images, there are detected calibration patterns with known size and shape in each view, so matching these points between the images is simple. After the matching, the intrinsic matrix can be computed, for example, by Zhang's method [14]. This matrix can then be used in photogrammetry to speed up and improve computation.

The second way of obtaining the intrinsic camera matrix is using optimization inside the photogrammetric software using bundle adjustments. This way, it is not needed to recalibrate the camera for each new scene where the camera focus changes. For convenience and similar results, this thesis uses this way.

2.3.2 Light Calibration

For working with the light sources in the virtual environment, it is crucial to obtain for each light source two pieces of information: light pose and light intensity characteristics.

Light Source Pose Estimation

To correctly place the light source into the virtual scene, it is needed to estimate the light source's position and orientation relative to the setup (or camera). This information can be obtained directly from the setup model when using the proposed construction, as the construction allows minimal movement. However, it needs to be optimized for variations of this setup where manual adjustments are made.

For calibration of a scene with an unknown position and orientation of the light sources, see Quéau et al. [15].

Light Intensity Calibration

In this setup there ale used LEDs to illuminate the scene. LED light source can be modeled as an imperfect point light source. It can be assumed that the area of the emitting element in the LED is small enough to be considered a point light source. On the top of the diode, a lens is often used to aim the light in a particular direction (most often

in the direction of the normal vector of the LED surface). This lens creates a relative intensity characteristic, causing the light's intensity to change with the emitting direction. Most often, the light intensity lowers with an angle from the normal to the surface of the diode. Such characteristic is called radial attenuation. The manufacturer usually measures the radial attenuation characteristic into an IES file or a graph in the LED datasheet. For general intensity characteristics calibration see [15].

The relative intensity characteristic is then scaled by a scalar factor denoting the actual intensity. The intensity is dependent on the current going into the LED and the effectivity of the led.

Measurements of the BRDF are done by observing pixel values in the image; therefore, the intensity scalar is measured relative to the sensitivity of the camera and the intensity of the light sources. Visual texture in this thesis is measured by the same high-power LED models with the same current and PWM settings using the same camera. As there are no other light sources in the scene, the intensity scalar can be omitted. This assumption causes that the intensity unit is a pixel value from 0 to 255, and the results are not comparable between different setups (different camera or light sources) without further calibration.

When the material is measured using multiple different LEDs, different LED configurations, or a different camera, the intensity scalar needs to be calibrated. This calibration can be done by observing a diffuse gray board and measuring a relative scaling factor between the intensities of the LED reflections. The relative scaling factor can then scale the measurements into unified units across multiple acquisition setups.

3 Proposed Method

The main goal of this thesis is to propose a method of visual texture measurement and material model estimation. This chapter describes the entire method. At first, there is a general description of the whole computational pipeline. After that, all steps are explained in detail with concrete choices of the used technologies.

3.1 Computation Pipeline

This section introduces a general pipeline of the visual texture reconstruction for photogrammetry scanning.

The first step of measurement is data acquisition. Two types of data are needed: images from each view with global ambient illumination and images from each view illuminated with each light source in the setup. Globally illuminated images are used for photogrammetry reconstruction as they contain minimal shadows. They can also be used to create object masks, speeding up the computation process. Illumination from separate light sources is then used for the BTF measurement. The full acquisition process is covered in Chapter 2.

The second step is mesh reconstruction and viewpoint estimation. In this step, a mesh is reconstructed (from a point cloud obtained by photogrammetry). This step also estimates the camera calibration and the view matrix for each object position.

The third step is the computation of BTF by projecting each pixel of each image onto the mesh with its corresponding acquisition information. In the fourth step, this information is then transformed into normalized orientation, so data from multiple images can be grouped.

The fifth step is sampling measurement points on the surface of the mesh to allow analysis and rendering of the material. In the sixth step, the measurements can be used for material model parameter estimation.

Material can then be validated in the seventh and last step by rendering and comparing it to the real image data.

3.2 Mesh Reconstruction and Viewpoint Estimation

The first step of mesh reconstruction is obtaining a point cloud of the captured scene from the acquired images. Pointcloud can be reconstructed in various ways. For this thesis, the COLMAP software has been used. It is open-source software for point cloud estimation. As a byproduct of its computation, it can also estimate each view's camera intrinsic matrix and view matrix.

For BTF mapping, the mesh is needed. It can be automatically estimated using Screened Poisson surface reconstruction, or, when previously known, the object's model can be fitted to the point cloud and used instead of the reconstruction. The latter method has superior results as it does not contain photogrammetry or surface reconstruction errors. However, the model is not known in many cases. This thesis uses Screen Poisson surface reconstruction implemented in MeshLab [16]. The reconstructed mesh from this method has some residuals of the fitted plane and observed surface surroundings. Therefore it has been, for computation speedup, manually removed as it would be removed from the BTF measurement by the object mask. For an example of a reconstructed mesh with and without manual cropping, see Fig. 3.1.

3.3 BTF Measurement

When a mesh is reconstructed, it is possible to map the measurement images (images from each viewpoint and with each light condition) onto the mesh. The mapping is done by casting rays through each image pixel into the scene. The ray casting is done using NVIDIA OptiX [17]. OptiX is a hardware-accelerated ray tracing engine. The engine allows casting rays through a scene efficiently and can hold data per a ray.

First, the setup (camera and light source) is transformed to the proper positions using the view matrix. Than, the ray is cast for each pixel in the direction

$$v = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{3.1}$$

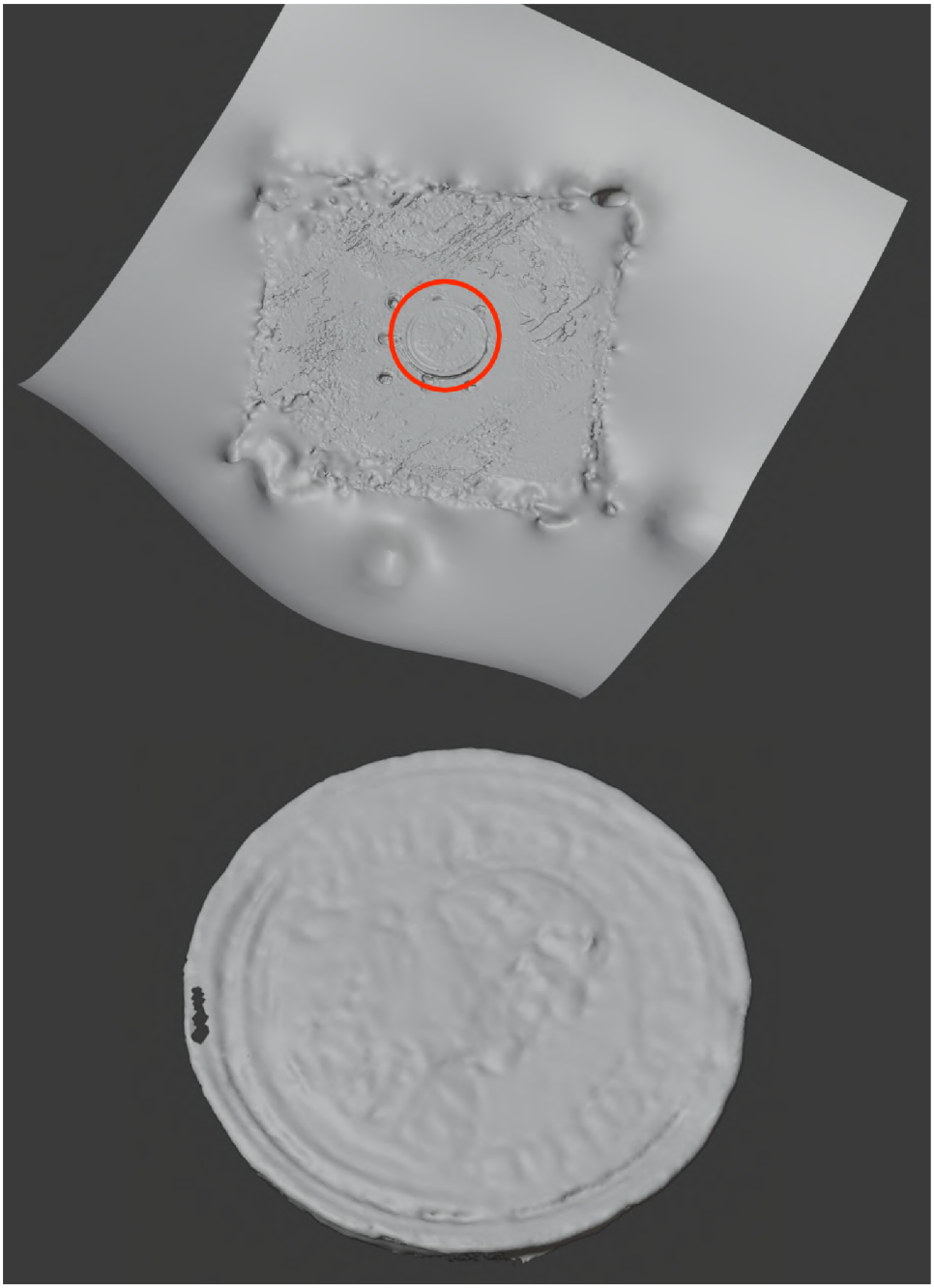

Figure 3.1: Comparison of the original reconstructed mesh (top) with the used part of the mesh (bottom).

where K is a camera matrix and u, v are pixel coordinates. When a ray is cast from a camera through a pixel, it is assigned a value of the pixel. The pixel value represents an amount of light acquired by the camera for each RGB channel (representing the three wavelengths). When the ray misses the mesh, it is discarded. When the ray hits the surface (at the hit point), it is first confirmed that a shadow does not occlude the hit point. It is done by casting a ray towards the light source, and if the it hits any mesh before it reaches the light source, the hit point is under a shadow. When the hit point is not under a shadow, it is assigned with the required scene geometry information. As light cast from a LED in a different direction has significantly different intensity, the value assigned to the ray must be corrected. The correction is done by dividing the value by the radial attenuation of the LED obtained from the calibration. Each valid ray after computation stores further data: intensity value, the normal of the surface, vector in the direction to the light source, distance from the light source, vector in the direction to the camera (eye vector), distance from the camera, hit point position, index of the triangle the hit point is in and barycentric coordinates inside the triangle. The BTF measurements are created by storing the data from all valid rays into an array.

3.4 Measurement Normalization

The BTF measurements are oriented differently in the space, complicating the material analysis and the material model estimation. The measurements can be normalized by rotating the measurement vectors (normal vector, light vector, and camera vector) so the normal vector would aim in the direction of a Z-axis. After that, there is no information available to obtain the correct rotation of the measurement around the Z-axis (normal), so isotropic BRDF (IBRDF) is assumed. When the isotropic material is considered, the azimuth of the eye vector is set to be zero, so the measurement can be rotated around Z-axis to align the eye vector with the XZ plane in the positive direction of the X-axis. After the rotation, the measurement vectors can be compressed into three numbers: the light vector's elevation θ_l, the light vector's azimuth ϕ_l, and the eye vector's elevation θ_e.

3.5 Measurement Sampling

BTF measurements from the previous step are non-uniformly placed on the surface of the mesh. There must be a way to assign a general point on the surface with its reflectance properties for further usage. It is done by sampling material representatives (points for which the material is computed) on the top of the surface. The most forward way is to choose all vertices as representatives. This approach might lead to a more sparse representation of larger triangles than the small ones. This thesis introduces a new texture mapping representation called Barytex forest to account for this issue.

Barytex forest is a texture sampling based on an iterative subdivision of the mesh triangles. Barytex forest is a forest of measurement trees, where each tree represents a sampling of a triangle of the observed mesh (observed triangle). The measurement tree consists of two parts: measurement points (points with material data sampled on the observed triangle) and a measurement tree which is a complete quaternary search tree.

Each node of the measurement tree represents a sub triangle inside the observed triangle. It stores the current triangle in the form of indices of three measurement points. At the root of the tree, there is the observed triangle. Each edge of the triangle is subdivided in the middle. The middle points connected by edges divide the root triangle into four smaller triangles. These four triangles form the first layer of the tree. The process is recursively repeated for each sub triangle until the stop condition is reached. The stop condition has been chosen with two criteria: the minimal edge length and the maximal tree depth. When the tree exceeds the maximal tree depth or if the minimal edge length is greater than the length of the largest edge of the smallest triangle, the subdivision stops. The tree is illustrated in the Fig. 3.2. After the subdivision, multiple points are created. These points are called measurement points, as they are further used as representatives of the BTF measurement. Tree leave triangles create a grid (texture grid). Material of any point inside the observed triangle (observed point) is computed by an interpolation of the measurement points of the texture grid triangle, inside which lies the observed point.

This allows to represent measurements of a material by uniform grid of samples without the need of UV unwrapping and material

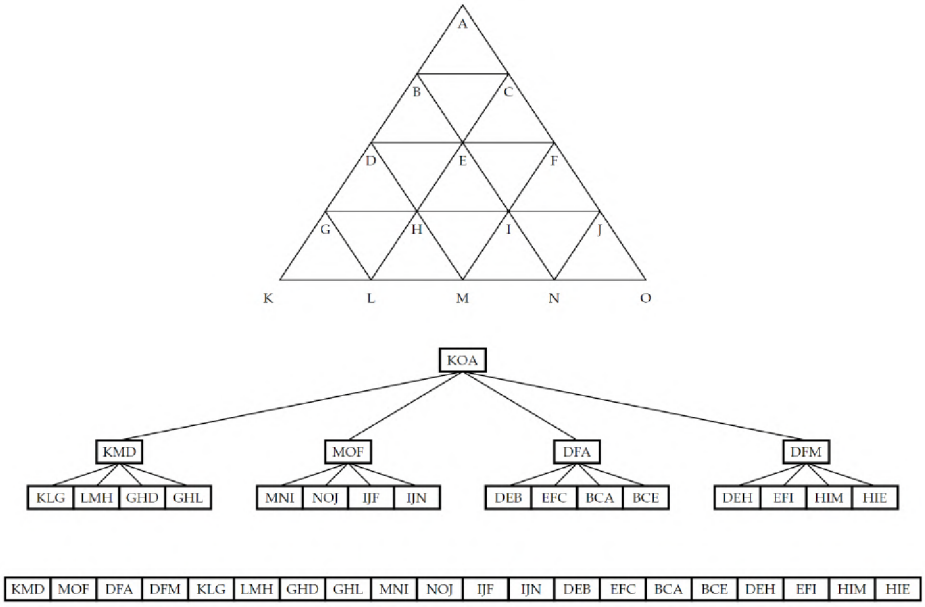

Figure 3.2: Material tree illustrations: triangle subdivision grid (top image), tree structure (middle image, each node represents a triangle), data alignment (bottom)

model creation. It is also good for raytracing rendering purposes as it provides fast searching of texture representatives and fast resolution lowering for triangles further from the camera to increase the rendering speed.

3.5.1 Barytex Forest Creation

Barytex forest is computed by separately creating the measurement trees. For each triangle of the observed mesh, a measurement tree is created. A measurement tree is represented by two arrays: measurement points and material tree nodes.

The first step is to allocate the space for the arrays. The sizes of the arrays are dependent only on the tree depth, which can be computed straight from the stop criteria. The number of subdivisions (tree depth)

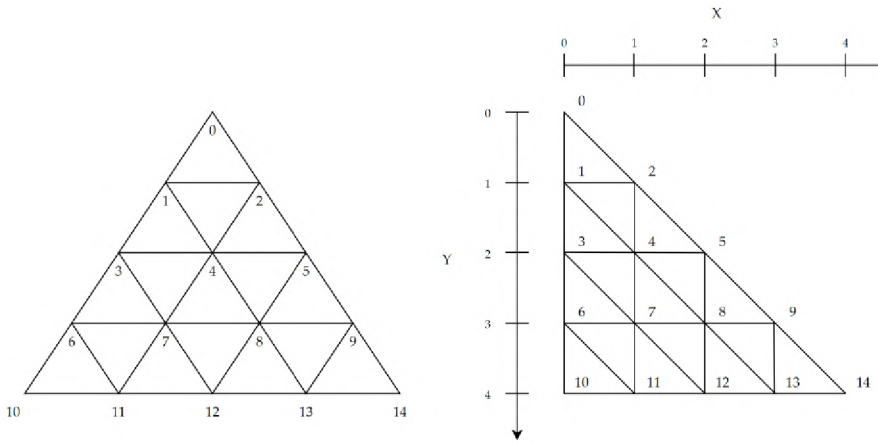

Figure 3.3: Ordering of the measurement points (left) and the 2D index mapping (right).

needed to reach the minimal edge length can be estimated as

$$d_e = log_2\left(\frac{a}{b}\right), \tag{3.2}$$

where a is a maximal edge length of the observed triangle and b is a minimal edge length provided by the user. Than the resulting tree depth is

$$tree_depth = min\left\{d_e, d_m\right\}, \tag{3.3}$$

where d_m is a user defined maximal tree depth.

As the measurement tree is a complete quaternary tree the number of tree nodes n is

$$n = 4^{tree_depth}. \tag{3.4}$$

When building the tree, multiple triangles refer to the exact measurement point, which might lead to redundancy of measurements. Therefore a proper ordering of the measurement points is needed to allow an easy search for already created measurement points. The measurement points with subdivision edges create a grid inside the observed triangle. The grid can be transformed (see Fig. 3.3) to allow the ordering of the measurement points in an array. After the transformation, it can be seen that there is always one more point in each

layer than in the previous layer. As the triangle edges are divided in half the layer count l is

$$l = 2^{tree_depth} + 1, \tag{3.5}$$

then the number of unique measurement points m is

$$m = \frac{l(l+1)}{2}. \tag{3.6}$$

Using the measurement point ordering from Fig. 3.3, the measurement point index can be mapped into 2D coordinates. This mapping allows finding the index of a middle measurement point between two known points by averaging the 2D coordinates and remapping the average point back to the 1D index.

The material tree nodes are ordered in the array as seen in Fig. 3.2. This ordering allows obtaining node positions in the tree, its parent nodes, and the child nodes (with its ordering) straight from the node index.

The first step of creating the measurement tree is creating the root node. The root node represents the observed triangle. Thus, the indices of its corresponding points are

$$\begin{aligned} i_A &= 1 + \frac{l(l-1)}{2} \\ i_B &= m - 1 \\ i_C &= 0. \end{aligned}$$

Measurement points at indices i_A, i_B, i_C are assigned corresponding world coordinates from the observed triangle and are marked as searched, creating the root node.

Next, the nodes are searched in their order in the array. The measurement points are created while creating the nodes. The used ordering assures that all triangle points represented by parents of the currently searched node are created and contain a valid position in space.

As all the subdivision steps are the same for each node's triangle, the term current triangle is used for the triangle of a currently searched node. The nodes are searched in the order in the array, starting from

the root. A measurement point is created for each pair of points of the current triangle (current point pair). The newly created measurement point index is computed by averaging the 2D grid mapping of the current point pair's indices and remapping the result back into an index. If the index is lower than m and the measurement point on the index is not searched, the measurement point is assigned the average of the world coordinates of the current point pair as its world coordinate and is marked as searched. After the measurement point creation, child nodes can be initialized. Newly created measurement points can divide the current triangle into four child triangles. Each of the four child nodes is assigned indices of its corresponding measurement points (see Fig. 3.2). This process repeats for each node until the tree is created.

3.5.2 Measurement Representative Searching

When a ray hits the observed triangle in a point (hit point), the ray has to be assigned a measurement value. The measurement value is assigned by interpolating the corresponding representative measurement points. The representative points are chosen to be measurement points of a triangle represented in the material tree node. The following process finds the representing node.

The search starts from the root. It is assumed that the hit point lies inside the observed triangle. Therefore, it lies inside the root triangle. Each node in the searched path approximates more precisely the representative triangle. When passing through a current node, the next node is the child node of the current node, which contains the hit point. Each child represents a sector of the current node (see Fig. 3.4). Therefore if a hit point lies in a child's sector, the child node is chosen to be the next in the search. Except for the middle sector, each child sector is created from a corner point of the current triangle and the points of the sector's divisor. A divisor is a line segment separating a corner sector from the middle one (illustrated as a border between sectors in Fig. 3.4). The hit point lies in the sector if it lies in the half-plane described by the divisor and the corner point. If the hit point does not lie in any corner sectors, it lies in the middle one. The process is repeated until reaching a leave node or exceeding a maximal searched depth specified by the user.

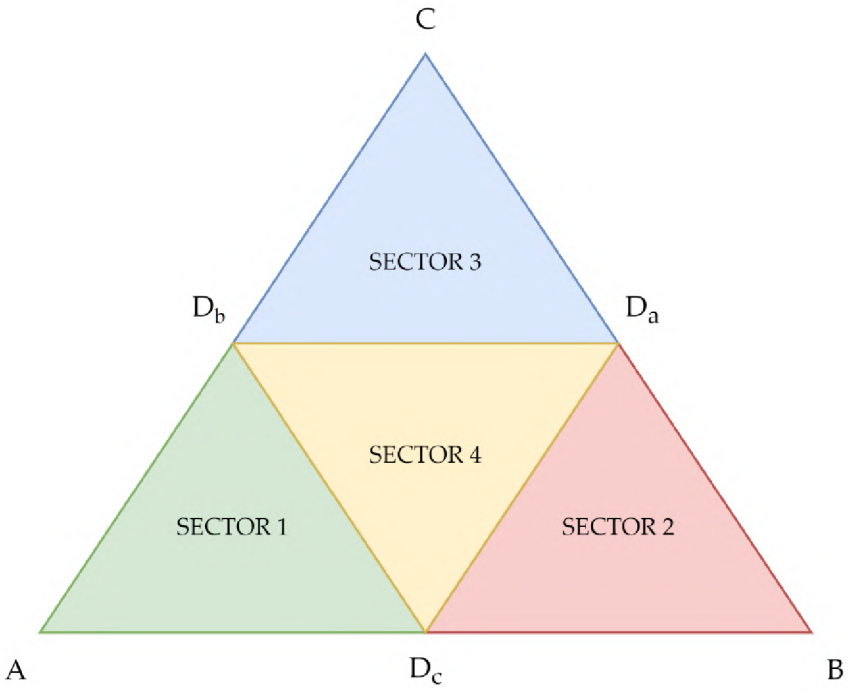

Figure 3.4: Visualisation of the measurement tree search sectors. Notation is as follows: current triangle ABC, divisor points D_a, D_b, D_c.

3.5.3 Valid Measurements Selection

As each image pixel from each camera position and each lighting can create a measurement, there can be a large number of measurements, concretely:

$$n_{mes} = n_{pix} \cdot n_{pos} \cdot n_{light}, \tag{3.7}$$

where n_{pix} is image pixel count, n_{pos} is the count of the view positions, and n_{light} is the count of the light sources. Which for setup used in this thesis leads to up to 3068928000 measurements. Note that the actual number is significantly lower (hundreds of millions, depending on the object) as the projection of the observed object does not fill up the whole image, and many measurements lie in a shadow, so they are discarded.

If all of the measurements were searched to find the proper measurements for each point sample, the computation would take an enormous amount of time. Therefore the measurements must be sorted, so only the measurements close to the processed sample point are searched. There are two possible approaches. The first is to order the measurements into a space search tree such as an octree or a k-d tree. This approach allows a fast search if the Euclidean distance is used as the distance metric. In a complex mesh, this metric could lead to a material mix-up as the closest measurement position to the sample point can be far on the surface of the object's mesh (see Fig. 3.5). As the measurement retrieval process allows to assign the triangle id to the measurement, it has been chosen to search for the closest measurement points by searching in the triangle where the sample is located, then its neighboring triangles. There has not been a need to search further as the measurements are densely sampled on the object's surface. However, the search process can be extended by a breadth-first search of the triangular grid until the needed amount of measurements is reached or the distance from the sample point is too large. Measurements found in the search process are then ordered by the distance from the sample point and the user-specified number of closest measurement (for this thesis, the first hundred measurements has been chosen). These measurements are than weighted by the inverse of the distance to the sample point and passed to the material solver.

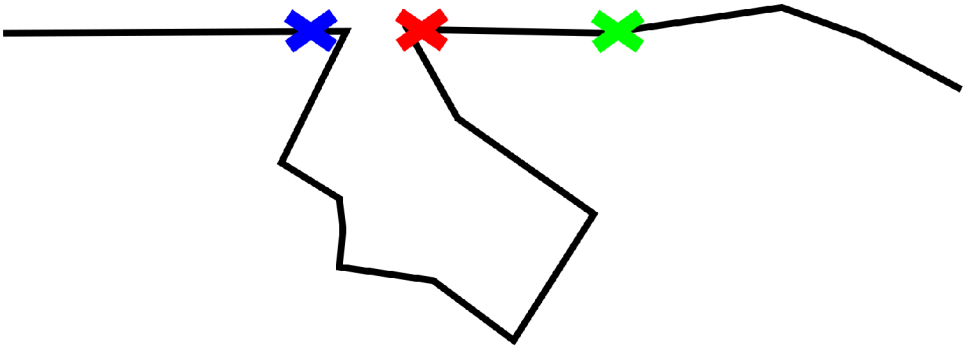

Figure 3.5: Visualisation of the error of Euclidean metric when choosing the best measurement for a sample point (red). The measurement on the blue point is the closest in space, but the measurement in the green point is better choice.

3.6 Material Model Computation

Previous steps provide a list of normalized IBRDF measurements attached to uniformly sampled points on the surface of the observed object. In this section we show how to use the measurement to compute material model parameters.

Material models are functions prescribing the material reflectance. This thesis uses the Lambertian reflectance function, Blin-Phong reflectance function, and the Cook-Torrance reflectance function. The most widely used optimizations of function parameters are based on gradient descent. The Ceres Solver library [18] has been chosen as an implementation of the gradient descent based nonlinear solver.

The reflectance model is computed separately for each color channel of each sample point. There has been used a least-squares loss function:

$$E = (Y_p - Y_o)^2, \tag{3.8}$$

where Y_p is the intensity value predicted by the model, and Y_o is the intensity value observed by the measurement. All the parameters of all the used models have to be non-negative, so the constrain was added to the solver. After the loss function choice, the nonlinear least squares optimization base on the Levenberg-Marquardt algorithm [19, 20] is done to obtain model parameters.

These parameters are then stored inside the measurement samples and stored in the form of Barytex forest material.

3.7 Validation Rendering

The validation of the measurements can be done by rendering the reconstructed mesh with the computed material. This thesis uses the NVIDIA Optix render engine to render the mesh straight from the Barytex forest. If a different render engine would be used, the mesh would have to be first unwrapped into the UV coordinates. Then the material parameters would have to be stored into discrete texture maps, which could change the material's properties and corrupt the validation process.

At first, the ray is cast, and the hit point (the point where the ray intersects the mesh) is estimated. Then the closest measurement triangle is found using the process described in Section 3.5. When valid measurement samples are retrieved, the material reflectance is computed for each measurement sample. These reflectances are then interpolated to obtain the material reflectance of the hit point. This is done for each camera ray resulting in the rendered image.

4 Evaluation and Results

This chapter introduces the validation dataset. Shows results of the proposed method. First, it validates each step of the method separately, and then it compares the results of the proposed method with actual data acquired with the setup and discusses the results.

4.1 Validation Dataset

The validation dataset used in this thesis consists of four objects. The first is the printed circuit board used in the acquisition setup (PCB) to represent an object with high details. Next, there is a rock object representing a mostly diffuse non-planar object. The third object is a coin, representing a metallic, high detail surface. The last one is a scratched magnesium box (box) representing a larger planar surface with a print and a few highly reflective scratches. All the objects can be seen in Fig. 4.1.

Each object is captured in 40 positions, and each position is captured under 37 light conditions. Images were purposely under-exposed to contain as much information in the highlights as possible. The shadows were not used in measurement, as the highlights are more important. An example of the input images is shown in Fig. 4.2.

Some of the images in the dataset contain two errors caused by the camera error. Future work can avoid these errors using a more expensive industrial camera, which was not available for the acquisition setup prototype.

First, the camera used in the setup had a hardware error on the bus, so some images contain an invalid sequence of pixels. This invalid sequence is impossible to recognize in the measurement process unless the sequence is entirely black. A sequence of colored invalid pixels does not occur often enough to corrupt the material model computation process but results in some invalid measurements.

The next error is in the white camera balance (which could not be fully turned off), where some images have unpredictable different color hues. This error can influence the texture measurement and the material model estimation.

Figure 4.1: Validation dataset: PCB (top left), rock (top right), coin (bottom left) and box (bottom right).

Figure 4.2: Example of input images for the rock object.

Figure 4.3: Visualization of the reconstructed mesh of the rock object.

4.2 Mesh Reconstruction and Measurement Validation

The first part of the evaluation is the evaluation of the BTF measurement process. The measurement is highly dependent on the reconstructed mesh and viewpoint estimation. Therefore the mesh reconstruction is evaluated first. Second, the viewpoint estimation can be evaluated. When the scene parameters and mesh are validated, the BTF measurement can be evaluated.

4.2.1 Mesh Reconstruction Validation

Reconstructed mesh must be validated by a visual comparison of the mesh to the real object as no ground truth mesh is available. Reconstructed meshes are shown in the Figures 4.3, 4.4, 4.5 and 4.6 (see Attachments for the reconstructed meshes).

All the reconstructed meshes are dense and contain the main features of the real objects but contain some errors. The errors can be grouped into two groups.

The first group of errors is high-frequency noise. It can be best seen on large flat surfaces (see Fig. 4.6). These errors are caused by

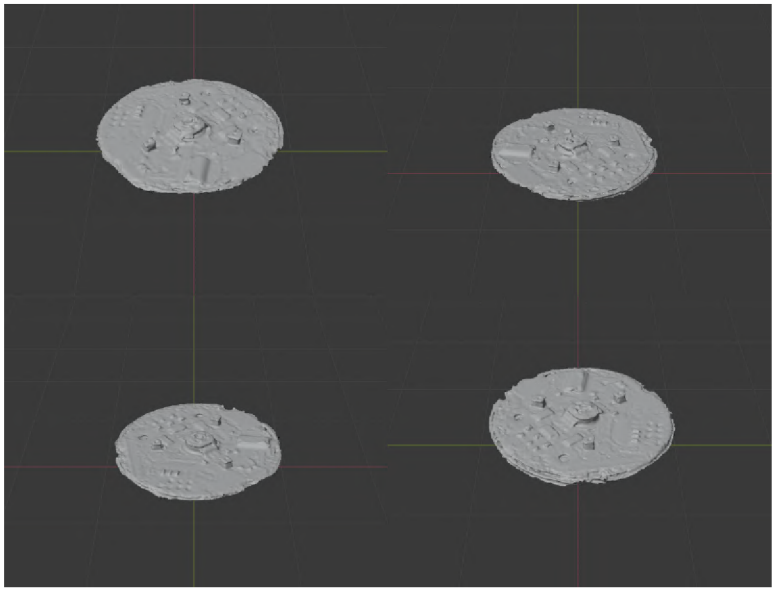

Figure 4.4: Visualization of the reconstructed mesh of the PCB object.

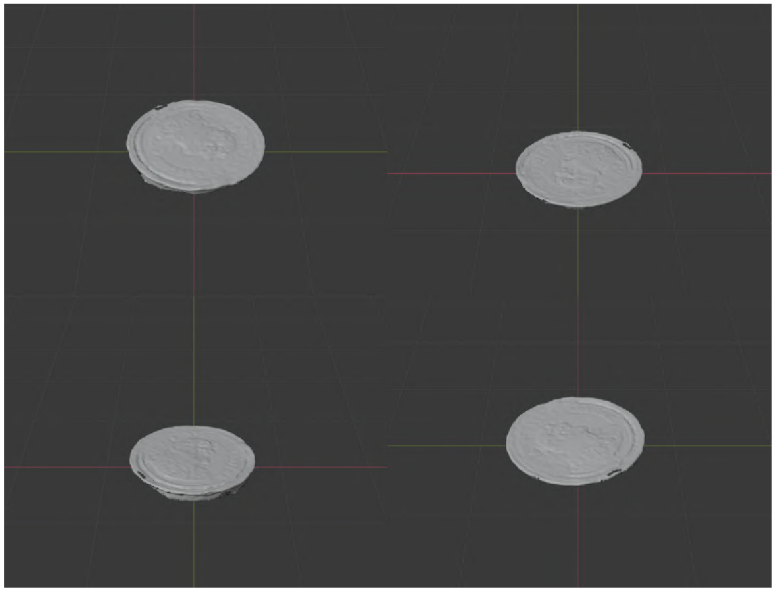

Figure 4.5: Visualization of the reconstructed mesh of the coin object.

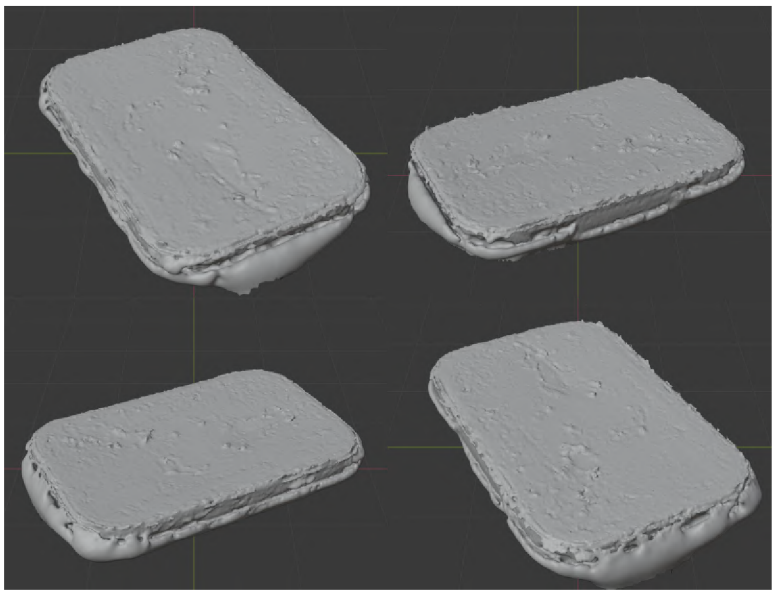

Figure 4.6: Visualization of the reconstructed mesh of the box object.

imprecise point cloud estimation, where the depth of points on a flat surface is estimated with a little error (up to lower tenths of a millimeter). This error is not significant for applications like object pose estimation, but it can significantly change normal vectors of points on the surface of the mesh.

The second group of errors is wrong mesh estimations around the sharp edges of the object. The sharp edges are approximated by the Screened Poisson reconstruction, where there are used B-Splines to fit the object's surface. These B-Splines do not have high enough frequency to fit the edge precisely, so they try to round the edge, but when point cloud outliers are created around the edge, the rounding can create an arbitrary fold of the surface. It can be seen in the Fig. 4.4 on the edges of the small LEDs or on the edges of the box in the Fig. 4.6. The surface folds might occlude a valid part of the mesh in many images and corrupt the BTF measurements of the occluded surface.

4.2.2 Viewpoint Estimation and BTF Measurement Validation

When the mesh is reconstructed, the viewpoint estimation and, with it, the BTF measurement can be evaluated. In this process, the most

Figure 4.7: Images of the coin object, captured from multiple viewpoints, overlaid with a red render of the reconstructed mesh.

crucial property is the precision of the estimated hit points. The position of each hit point must be an exact projection of a pixel onto the reconstructed mesh to allow proper mapping of the measurement value.

The precision of the viewpoint estimation can be measured by rendering the mesh from an estimated viewpoint and its comparison to the real image acquired from the viewpoint. Real images with overlaid renders of the mesh are shown in Fig. 4.7, where it can be seen that the renders fit precisely into the images, except for the errors in the mesh estimation.

Measurements can also be shown in the form of a colored point cloud to examine the density of the sampling (see Fig. 4.8). The distance between the measurement samples from a single viewpoint on a surface parallel to the image plane (largest sample density) is for all the dataset objects around 0.01 mm. The density further increases with the number of viewpoints used in the measurement.

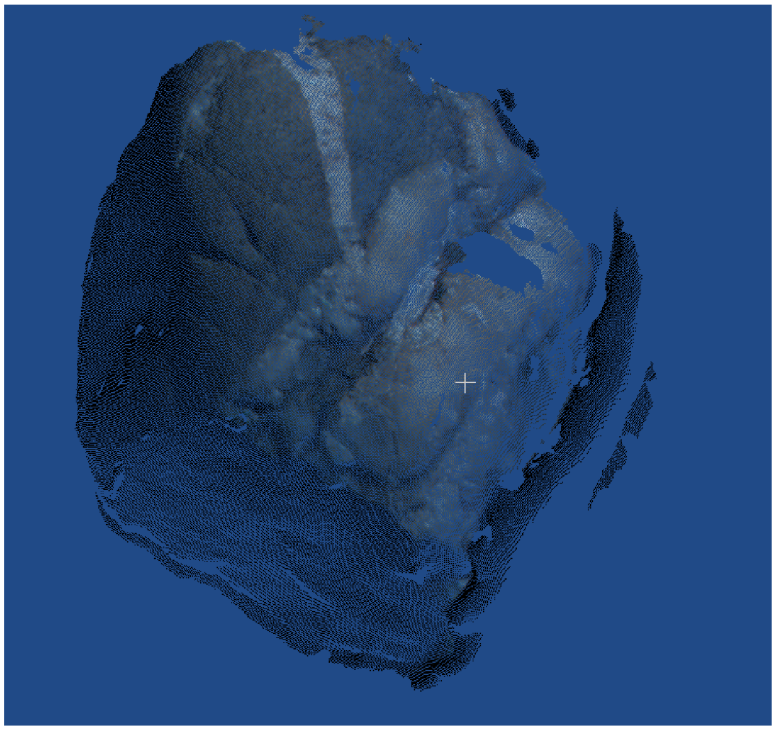

Figure 4.8: Measurements obtained from a single image of the rock object in the form of a colored point cloud.

Table 4.1: Table of mean absolute errors for each object form the dataset.

	Lambert	Blin-Phong	Cook-Torrance
coin	0.02979	0.03074	0.21930
rock	0.02339	0.02470	0.12989
PCB	0.04383	0.04364	0.19217
box	0.03743	0.03845	0.15958

4.3 Validation of Material Model Estimation

The last part of the validation process is evaluation and validation of the material model estimation. For this thesis were used three material models: Lambertian reflectance model, Blin-Phong reflectance model and Cook-Torrance reflectance model. For each object from the dataset, material models has been estimated from measurements acquired from a fifth of the acquired images. The objects are then rendered using the estimated material from viewpoints and light conditions unused for the measurements. These rendered images are than compared to the images acquired from the same viewpoints and light conditions. Comparison has been done in two ways: mean absolute error (MAE) between renders and real images and visual observation.

The mean absolute error is computed as

$$E = \frac{\sum_{i=0}^{N}\sum_{j=0}^{K}\sum_{p}^{\Omega} M_i(p)|R_{i,j}(p) - I_{i,j}(p)|}{\sum_{i=0}^{N}\sum_{j=0}^{K}\sum_{p}^{\Omega} M_i(p)}, \tag{4.1}$$

where N is the number of viewpoints, K is the number of light sources, p is a pixel index from image domain Ω, M_i is the mask of object in the view i and $R_{i,j}$,$I_{i,j}$ are rendered image and real image from viewpoint i under light source j.

Mean absolute error for each material model on each object from the dataset can be found in Table 4.1. For error images and MAE for each image separately see the Attachments.

After the MAE evaluation a visual validation was done. Comparison of real image to a render image for each dataset object using each reflectance model can be seen in Figures 4.9, 4.10 and 4.11. For render

images from more viewpoint and under more light conditions see the Attachments. Further there are discussed performances of the three used reflectance models.

The Lambertian reflectance model (LRM), as the only linear reflectance model used, had the fastest and the most stable parameter estimation. As seen in the comparison images (see Fig. 4.9) the model represents well the colors and base material features, but lacks any specular highlights, because it can represent only the ideally diffuse materials. The next problem of this model is that it might leave some artefacts in the resulting albedo. These artefacts are residuals of highlights from the measurement (see the PCB object in Fig. 4.9).

The Blin-Phong reflectance model (BPRM) as an extension of the LRM, therefore it yealds similar results on diffuse surface (with same advantages and disadvantages). The difference is that , using the specularity term, it can introduce a directionality to the reflection resulting in a slightly better contrast of the renders (see rock and coin objects in Fig. 4.10). As the results of LRM and BPRM are similar see the Attachments for higher resolution and more images. The main problem of using this model is that the optimization process on some positions on the surface created a narrow strong specularity lobe causing a color noise on some render images (see coin in Fig. 4.10).

The Cook-Torrance reflectance model is a highly nonlinear function without a continuous derivatives, as such the proposed optimization process did not solve the model parameters on the majority of the surface. The optimization is mostly stable on synthetic data, but unstable for real data as they can contain noise. Other reason for this instability might be errors from the mesh reconstruction step. Wrongly estimated normal vector of the surface can change the measurements, thus corrupting the model estimation process. Due to the instability the rendered images are heavily corrupted by a color noise. For comparison of the rendered image to the real image see Fig. 4.11. For further usage a more stable reflectance model is recommended, such as high order polynomial approximations.

Figure 4.9: Comparison of real images to the rendered images (left) using estimated Lambert reflectance model (right).

Figure 4.10: Comparison of real images to the rendered images (left) using estimated Blin-Phong reflectance model (right).

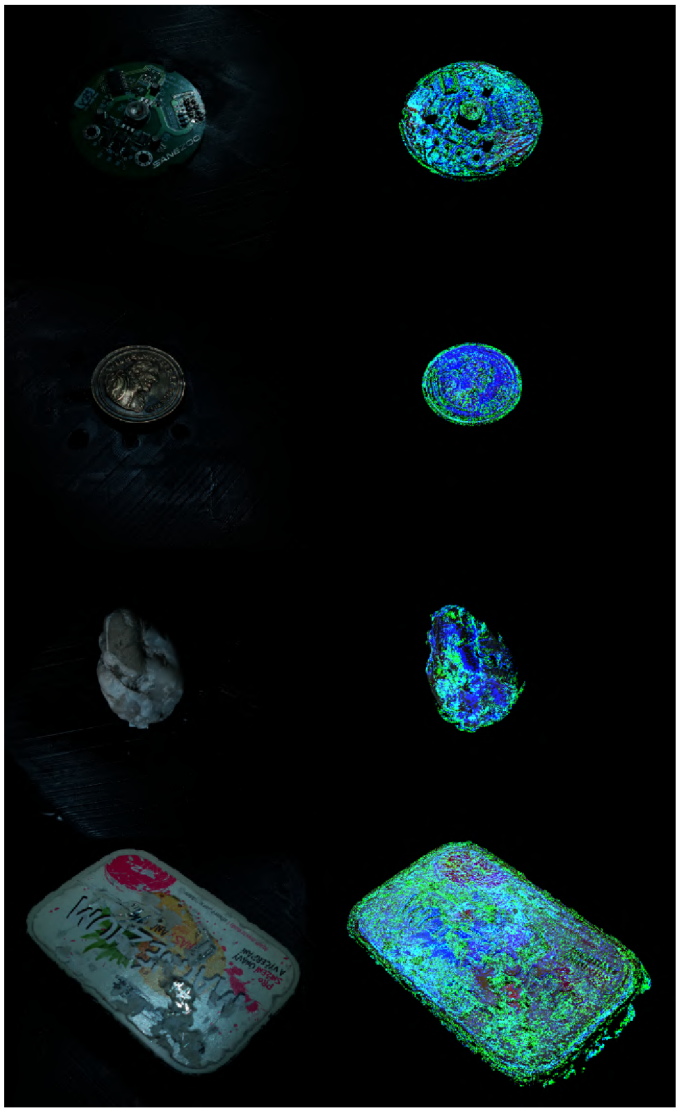

Figure 4.11: Comparison of real images to the rendered images (left) using estimated Cook-Torrance reflectance model (right).

5 Conclusion

This thesis proposed a pipeline for a visual texture measurement of real objects using images acquired from different positions under different light conditions. The acquired measurements can then be used for material analysis and rendering from general viewpoints and light conditions.

We designed, assembled, and calibrated a cost-effective acquisition setup based on a light stage and used it to acquire a validation dataset consisting of images of multiple different objects.

We reconstructed the mesh of the observed objects using a combination of photogrammetry and Screened Poisson surface reconstruction from acquired images.

We proposed a method of measuring reflectance properties by inverse rendering using NVIDIA OptiX rendering engine. Further, we introduced a new measurement sampling algorithm to cover the mesh uniformly with measurements. This algorithm uses a tree data structure called Barytex forest, which allows fast searching of material properties of a general point on the surface of the mesh. When used in rendering, it will enable simple runtime changes in the resolution of the material measurement to increase the speed of rendering materials of objects far from the camera.

We have shown a method of material model estimation from the measured data and its usage in rendering. We used the acquisition setup to acquire a dataset consisting of multiple different objects. Evaluated the proposed pipeline and computed the material models. We then validated models by comparing renders of the dataset objects against real images captured from the same viewpoint under the same light condition as those used in the rendering process.

5.1 Future Work

In the near future, we want to use the information from the prototype acquisition setup to create a more precise and more general acquisition setup, as many of the material model estimation problems might have been caused by an unstable dataset.

We want to use the measurements to increase the precision of the mesh reconstruction using photometric stereo to compute proper normals and then integrate them to compute the displacement of the mesh's vertices. The displacements can be used to analyze errors of the given mesh, which can be used in quality inspection, providing that the mesh of the ideal object is known.

The next exciting usage of the obtained measurement is material analysis and segmentation, which can then divide the mesh into parts with the same material. When a previous BRDF dataset is measured, we can recognize the material from the measurements. The recognition can be used, for example, in rendering or mesh improvement.

A Appendix

A.1 Acquisition Setup Images

Figure A.1: Acquisition setup with active blue global illumination.

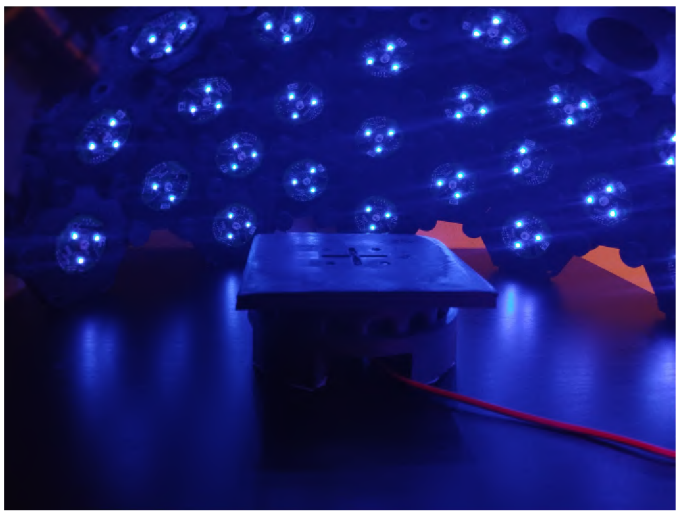

Figure A.2: Inside view of the acquisition setup with active blue global illumination.

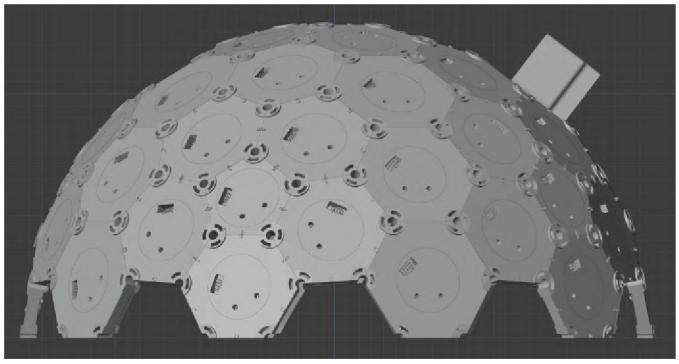

Figure A.3: Model of the acquisition setup from the side view.

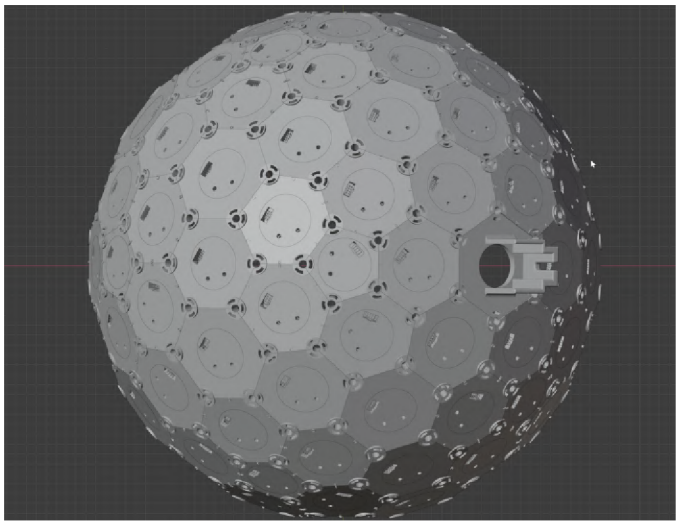

Figure A.4: Model of the acquisition setup from the top view.

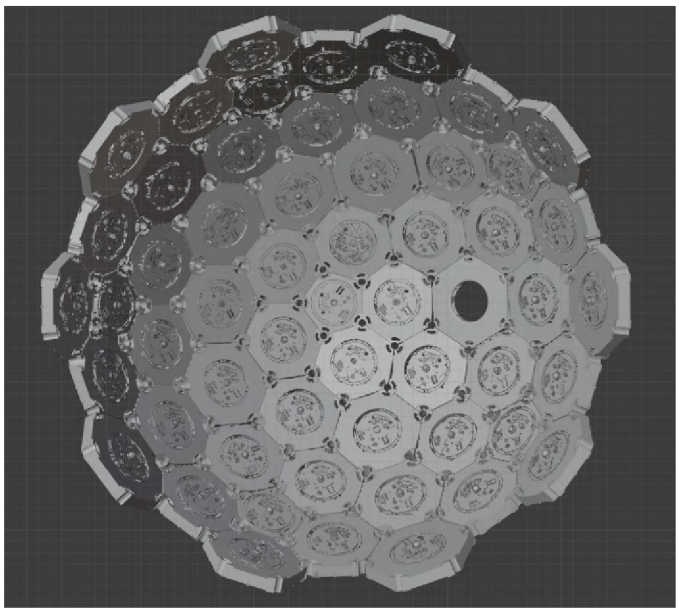

Figure A.5: Model of the acquisition setup from the inside.

Bibliography

1. ERB, Wolfgang. Computer-controlled gonioreflectometer for the measurement of spectral reflection characteristics. In: OSA, 1980, vol. 19, pp. 3789–3794. No. 22. Available from DOI: 10.1364/AO.19.003789.
2. DEBEVEC, Paul; HAWKINS, Tim; TCHOU, Chris; DUIKER, Haarm-Pieter; SAROKIN, Westley; SAGAR, Mark. Acquiring the Reflectance Field of a Human Face. In: 2000, vol. 2000, pp. 145–156. Available from DOI: 10.1145/344779.344855.
3. HAINDL, Michal; FILIP, Jiří. *Visual Texture*. 2013. ISBN 9781447149019. Available from DOI: 10.1007/978-1-4471-4902-6.
4. COOK, Robert; TORRANCE, Kenneth. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.* 1982, vol. 1, pp. 7–24. Available from DOI: 10.1145/965161.806819.

5. BECKMANN, Petr; SPIZZICHINO, André. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Pergamon P., 1968. Artech House Radar Library. ISBN 9780835742313. Available also from: https://books.google.cz/books?id=nn92AAAACAAJ.

6. SCHLICK, Christophe. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*. 1998, vol. 13. Available from DOI: 10.1111/1467-8659.1330233.

7. SCHÖNBERGER, Johannes Lutz; FRAHM, Jan-Michael. Structure-from-Motion Revisited. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

8. SCHÖNBERGER, Johannes Lutz; ZHENG, Enliang; POLLEFEYS, Marc; FRAHM, Jan-Michael. Pixelwise View Selection for Unstructured Multi-View Stereo. In: *European Conference on Computer Vision (ECCV)*. 2016.

9. SCHÖNBERGER, Johannes Lutz; PRICE, True; SATTLER, Torsten; FRAHM, Jan-Michael; POLLEFEYS, Marc. A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval. In: *Asian Conference on Computer Vision (ACCV)*. 2016.

10. BARNES, Connelly; SHECHTMAN, Eli; FINKELSTEIN, Adam; GOLDMAN, Dan B. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*. 2009, vol. 28, no. 3.

11. BERNARDINI, Fausto; MITTLEMAN, Joshua; RUSHMEIER, Holly; SILVA, Cláudio; TAUBIN, Gabriel. The Ball-Pivoting Algorithm for Surface Reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*. 1999, vol. 5, pp. 349–359. Available from DOI: 10.1109/2945.817351.

12. KAZHDAN, Michael; BOLITHO, Matthew; HOPPE, Hugues. Screened Poisson Surface Reconstruction. In: 2006, vol. 32, pp. 61–70. Available from DOI: 10.1145/1281957.1281965.

13. KAZHDAN, Michael; BOLITHO, Matthew; HOPPE, Hugues. Poisson surface reconstruction. In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. 2006, vol. 7.

14. ZHANG, Zhengyou. Flexible camera calibration by viewing a plane from unknown orientations. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 1999, vol. 1, 666–673 vol.1. Available from DOI: 10.1109/ICCV.1999.791289.

15. QUÉAU, Yvain; DURIX, Bastien; WU, Tao; CREMERS, Daniel; LAUZE, François; DUROU, Jean-Denis. LED-based Photometric Stereo: Modeling, Calibration and Numerical Solution. *CoRR*. 2017, vol. abs/1707.01018. Available also from: http://arxiv.org/abs/1707.01018.

16. CIGNONI, Paolo; CALLIERI, Marco; CORSINI, Massimiliano; DELLEPIANE, Matteo; GANOVELLI, Fabio; RANZUGLIA, Guido. MeshLab: an Open-Source Mesh Processing Tool. In: *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. ISBN 978-3-905673-68-5. Available from DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

17. PARKER, Steven; BIGLER, James; DIETRICH, Andreas; FRIEDRICH, Heiko; HOBEROCK, Jared; LUEBKE, David; MCALLISTER, David; MCGUIRE, Morgan; MORLEY, R.; ROBISON, Austin; STICH, Martin. OptiX: A General Purpose Ray Tracing Engine. *ACM Trans. Graph.* 2010, vol. 29. Available from DOI: 10.1145/1778765.1778803.

18. AGARWAL, Sameer; MIERLE, Keir; TEAM, The Ceres Solver. *Ceres Solver*. 2022. Version 2.1. Available also from: https://github.com/ceres-solver/ceres-solver.

19. KQ, Levenberg. A Method for The Solution of Certain Non-Linear Problem in Least Squares. *Quarterly Journal of Applied Mathmatics*. 1944, vol. 2, pp. 164–168.

20. DONALD W., Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameter. *Journal of the Society for Industrial and Applied Mathematics*. 1963, vol. 11, pp. 431–441.