REAL TIME TEXTURE MAPPED 3D RECONSTRUCTION USING A SETUP WITH MIRRORS AND CONTROLLED LIGHTING

A Thesis Submitted to

the Graduate School of Engineering and Sciences of

İzmir Institute of Technology

in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Electronics and Communication Engineering

by

Barış YAZAR

December 2021

İZMİR

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor Assoc. Prof. Dr. Şevket GÜMÜŞTEKİN for his guidance, support, and motivation throughout my studies.

Besides my advisor, I would like to thank the rest of my thesis committee members Assist. Prof. Dr. Mehmet Zübeyir ÜNLÜ and Prof. Dr Enver TATLICIOĞLU for their valuable comments and constructive criticism.

I would like to express my sincere gratitude to my colleagues at IZTECH for their continuous support and encouragement during tough times in the project. Lastly, I would like to thank my family for their endless support and love.

ABSTRACT

REAL TIME TEXTURE MAPPED 3D RECONSTRUCTION USING A SETUP WITH MIRRORS AND CONTROLLED LIGHTING

The purpose of this thesis is to create a 3D reconstruction framework that can be used in real time. This is accomplished using a parallel implementation of a shape from silhouette (SFS) algorithm. The number of silhouettes employed in reconstruction makes a major contribution to the quality at the expense of reduced speed. In order to keep this number at the minimum level without extensively sacrificing quality, a novel system is introduced. This system is based on evenly distributed viewpoints using a regular tetrahedron structure. In order to reduce cost and simplify camera calibration, we used a single camera setup with three mirrors thus creating virtual cameras for three of four viewpoints. Besides taking advantage of minimal number of viewpoints, parallel hardware is utilized to achieve real time speed. A volume based SFS algorithm is implemented using CUDA parallel computing platform.

ÖZET

AYNALAR VE KONTROLLÜ AYDINLATMA İÇEREN DÜZENEK KULLANARAK GERÇEK ZAMANLI DOKU HARİTALI 3B GERİÇATIM

Bu tezin amacı, gerçek zamanlı olarak kullanılabilecek bir 3B geriçatım sistemi oluşturmaktır. Bu, siluetten şekil elde etme algoritmasının paralel uygulaması kullanılarak gerçekleştirilmektedir. Geriçatımda kullanılan siluet sayısı, hızı düşürme pahasına kaliteye büyük katkı sağlar. Kaliteden büyük ölçüde ödün vermeden bu sayıyı minimum seviyede tutabilmek için yeni bir düzenek tanıtılmıştır. Bu düzenek, düzenli tetrahedron yapısı kullanarak eşit olarak dağıtılmış bakış açılarına dayanmaktadır. Maliyeti düşük tutmak ve kamera kalibrasyonunu basitleştirmek için üç aynalı ve tek kameralı bir düzenek kullanarak, dört bakış noktasından üçü için sanal kameralar oluşturulmuştur. Minimum sayıdaki bakış açısı avantajının yanı sıra, gerçek zamanlı hıza ulaşmak için paralel donanım kullanılmıştır. CUDA paralel hesaplama platformu kullanılarak hacim tabanlı bir siluetten şekil elde etme algoritması uygulanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF ABBREVIATIONS

SFS ……...……Shape from Silhouette

API……..……Application Programming Interface

IDE ……...…...Integrated Development Environment

GUI ……..Graphical User Interface

SDK ……..Software Development Kit

GPU ……...Graphics Processing Units

CHAPTER 1

INTRODUCTION

3D reconstruction, which is one of the main research areas in computer vision, is the process of acquiring 3D information from images or videos. Many applications, such as medical imaging, robotics, film industry, virtual reality, and augmented reality, depend on 3D reconstruction (Siudak at.al,2014).

There are two types of 3D reconstruction methods: passive and active (Butime et.al, 2006). Active methods, that are beyond the scope of this thesis, interact with the object by projecting some type of energy, such as lasers or ultrasound waves, onto the object, whereas passive methods do not require interaction with the object. Some passive methods of 3D reconstruction, also called Shape of X, are listed below (Szeliski, 2010).

- Shape from stereo-vision
- Shape from motion
- Shape from silhouette
- Shape from shading
- Shape from texture

The shape from stereo-vision is based on triangulation of corresponding points in the images taken with two cameras. Shape from motion takes advantage of camera movements. Shape from shading uses constraints in lighting, and shape from texture makes use of the variation in texture elements. Some of these methods take a lot of time, while others are difficult to implement.

Shape from silhouette (SFS) methods make use of the silhouettes of the object. SFS is the simplest method to get 3D information. It is one of the efficient and robust methods when compared to others. Also, it can provide useful initial solutions that can be refined by other shape from X methods (Kolev et.al, 2012). However, SFS generally produces low quality results and cannot fully reconstruct non-convex shaped objects. SFS can be computationally expensive if too many silhouettes with high resolution are used.

SFS methods are generally divided into two as volume and surface based methods (Siudak at.al,2014). Volume based methods are simple but computationally expensive. They are suitable for parallel computing. Surface-based methods are not simple, but they give results with extended details.

There are many studies and many setups related to SFS in the literature, but none of these studies take advantage of evenly distributed viewing angles. In this thesis, we propose and implement this setup with a camera and 3 mirrors, taking the corners of the tetrahedron as a reference, and examine its effect on the quality of the SFS results.

Camera calibration is an essential requirement for 3D reconstruction. In this study, a two stage approach is preferred. Camera intrinsic parameters are calibrated with a planar calibration object (Scaramuzza et.al 2006-a, Zhang, 2000). After the intrinsic parameters are determined, a spherical calibration object is used to get extrinsic parameters because it can be seen from all viewing angles (Guan et.al, 2015). When the system setup needs to be calibrated, it will be sufficient to perform only the second phase of the calibration. This provides flexibility for practical use.

Final objective of this thesis is to implement an SFS algorithm with a parallel computing acceleration to produce result in real time. To reach our goal, a volume based SFS algorithm is used due to its parallelizable nature, and CUDA library is utilized.

This thesis consists of 6 chapters, and its outline is as follows:

- Chapter 2 provides background information about used hardware, some open-source libraries, and software.
- Chapter 3 gives brief information about the setups that are used in literature and introduces optimal system and its unique features. This chapter also presents a novel design that was inspired by the optimal system.
- In Chapter 4, the camera models used in this thesis are described. Relevant camera calibration approaches are briefly reviewed. Two stage calibration algorithm implemented for our setup is presented.
- In Chapter 5, the types of SFS algorithms are presented, their pros and cons are discussed, and the proposed method is introduced.
- Chapter 6 summarizes the study with final remarks.

CHAPTER 2

BACKGROUND

In this chapter, background information about the hardware and software that were used in this thesis is provided. Firstly, electronic hardware such as a camera, piezoelectric sensor, and microcontroller is mentioned. Then, the software tools and libraries that were used are discussed.

2.1. Hardware

This section will provide comprehensive information on the hardware that used in the thesis.

2.1.1. Camera

In this thesis, the acA1300-gc gigE camera from the Basler company, shown in the Figure 2.1, is used. This camera has a Sony ICX445AQ CCD sensor. The Sony sensor has a width of 4.9 mm and a height of 3.6 mm. The pixels of the sensor are square, with a side length of 3.75 μm. This camera, which has a resolution of 1278x958 pixels, can capture 30 images per second.

The camera is used with a 12mm focal length lens which can be seen in Figure 2.2.

$$Field\ angle\ of\ view\ (degree) = 2 * Arctan(\frac{Sensor\ width}{2 * Focal\ Length}) \quad \textbf{(2.1)}$$

Figure 2. 1. Basler acA1300-GC gigE camera

According to equation 2.1, the camera offers a 23.1° field of view. This angle is insufficient for the intended setup with an object and three mirrors which are 20 cm away.

Figure 2. 2. Lens with 12 mm focal length

A Nikon FC-E8 fisheye adapter is utilized in addition to the lens for the reasons described above. Figure 2.3. shows the adapter. This adaptor transforms the lens into a wide angle fisheye format, as the name suggests, and reduces the focal length of the lens on which it is attached by a factor of 0.21.

Figure 2. 3. Nikon Fisheye converter.

2.1.2. Trigger Sensor

In this study, it is necessary to take a snapshot of a falling object. A sensor is needed to take a picture of the object at the right moment. A falling object can be detected with different types of sensors. Examples of these are laser sensors, sonic sensors, and piezoelectric film sensors. Piezoelectric film sensors that can be seen in Figure 2.4, are used in this study to detect a falling object in the right position.

Piezoelectric materials are good at converting energy between mechanical and electrical domains, even on the micro scale (Kanno,2009). The sensors which contain piezoelectric materials are rectangular elements of the piezo film with silver ink screen printed electrodes. These sensors are used primarily as dynamic strain gages and contact microphones for vibration or impact detection .

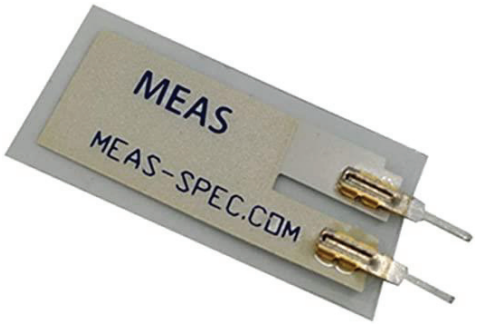

Figure 2. 4. Piezoelectric film sensor

2.1.3. Microcontroller

In our system, a microcontroller is required to establish communication between the sensor and the camera. In this work, Digispark ATtiny85 shown in Figure 2.5 is used.

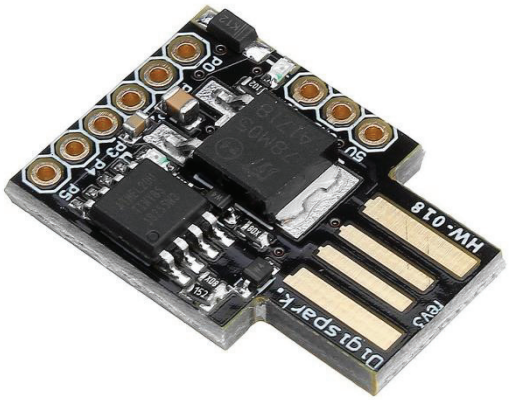

Figure 2. 5. Digispark ATtiny85 USB microcontroller

The Digispark ATtiny85 is a coin-sized microcontroller. It can be easily programmed with a computer via a USB port. It is a high-performance, low-power 8-bit microcontroller that can handle many tasks despite its small size. Some of its important features are listed below.

- 10-bit ADC
- 8-bit Timer/Counter with Prescaler and Two PWM Channels
- 8-bit High Speed Timer/Counter with Separate Prescaler
- 2 High Frequency PWM Outputs with Separate Output Compare Registers
- Six Programmable I/O Lines
- External and Internal Interrupt Sources

2.1.4. 3D Printer

In this study, many objects are designed to hold LEDs that are necessary for lighting and the mirrors. To print these objects, Ultimaker 2 3D printer is used. The designed objects will be shared in Appendix A.

2.2. Software

Several software tools and libraries were used in different stages of this work. Blender software is used to create simulated images. MATLAB is used for camera calibration. Visual Studio Toolbox is used with Plylon, CUDA and VTK libraries to develop the software.

2.2.1. Blender

Blender is an open-source 3D content-creation program that was created by The Blender Foundation, a nonprofit organization. It supports the entirety of the 3D pipeline, such as modeling, rigging, animation, simulation, rendering, compositing, and motion tracking, even video editing and game creation (Blender, 2021). It also allows users to design new tools using the Blender application programming interface (API) in Python.

Blender can provide an environment in which mirrors can be used as seen in Figure 2.6. All objects designed with Blender within the scope of this thesis are shared in Appendix A.

2.2.2. MATLAB

This software package has been widely used by engineers and scientists to analyze data, develop algorithms, and create models. MATLAB allows matrix manipulations,

plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Figure 2. 6. An example image with mirrors.

MATLAB has a high-performance language. It combines computation, visualization, and programming into a user-friendly interface where problems and solutions are written in familiar mathematical notation.

A vast amount of user supplied scientific code segments are available for MATLAB. Scaramuzza's fisheye calibration library (Scaramuzza et.al, 2006-a) can be given as an example of codes was used in this thesis.

2.2.3. Microsoft Visual Studio

Visual Studio is an Integrated Development Environment (IDE) developed by Microsoft. It's a software that allows programmers to write and edit code for GUI (Graphical User Interface), console, web applications, online apps, mobile apps, cloud, and web services, among other things. It contains completion tools, compilers, and other features to facilitate the software development process.

Microsoft Visual Studio was employed because C and Python languages were used throughout the production of this thesis in addition to C++. It can also be easily integrated with the CUDA library, which was used to enable parallel processing.

2.3. Libraries

The open-source libraries Pylon, CUDA, and VTK, were used in the development of the software.

2.3.1. Pylon Software Development Kit

The Pylon C++ API provides an interface to the Basler camera. It provides memory handling functions and camera specific control functions.

2.3.2. CUDA

CUDA was created by Nvidia. It is a parallel computing platform and API that enables software to use specific types of graphics processing units (GPUs) for general-purpose processing. CUDA is a software layer that allows computing kernels via direct access to the GPU's virtual instruction set and parallel computational units.

To build and deploy an application, the toolkit provides GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime library. Using the CUDA Toolkit, developers can create, build, and deploy programs on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, and cloud-based platforms.

There are other general-purpose computations on GPUs such as OpenCL and DirectCompute, but CUDA has several advantages over traditional general-purpose

computations on GPUs using graphics APIs. Some of them are given below (CUDA, 2021).

- Scattered reads – code can read from arbitrary addresses in memory.
- Unified virtual memory
- Unified memory
- Shared memory
- Faster downloads and readbacks to and from the GPU
- Full support for integer and bitwise operations, including integer texture lookups

In addition to the advantages described above, CUDA makes parallel programming easy. On the other hand, the biggest disadvantage of CUDA is that it can only work with NVIDIA brand GPUs.

This study is implemented using the CUDA platform. Parallel programming is used to do all the calculations between image acquisition and visualization that are preprocessing, point cloud processing.

2.3.3. Visualization Toolkit

The Visualization Toolkit (VTK), which is part of Kitware's collection of commercially supported open-source platforms for software development, is a freely available software system for image processing, 3D graphics, volume rendering, and visualization. It can handle a variety of visualization algorithms as well as advanced modeling approaches. VTK is cross-platform and runs on Linux, Windows, Mac, and Unix platforms (VTK, 2021).

VTK has been used in this study to visualize the 3D point cloud, and the reconstruction object.

CHAPTER 3

SYSTEM DESIGN

In this chapter, several setup designs suitable for SFS based 3D reconstruction are discussed.

Firstly, some important aspects of the design are introduced. Some design types are reviewed in the literature. Finally, the proposed designs are described and highlighted.

3.1. Design Aspects

In this section, three elements of the design, which are budget, number of snapshots, and distribution of the viewing angles are discussed.

The first element of designing a typical imaging system is the budget. For the SFS system, increasing or decreasing the number of cameras tremendously affects the budget. Using other equipment such as step motors, special tables, and simultaneous trigger mechanism also increases the project's expenses. Designers need to lay out the optimum and preferably the most inexpensive system to reach end-users.

Another important design parameter is the number of snapshots. Theoretically, perfect reconstruction is possible with an infinite number of snapshots in all view angles. However, it is not achievable in real life. Some setups take lots of images (Cheung et.al, 2003), while others use less than four (Forbes et.al, 2004). A large number of the snapshots could mean precise reconstruction, but it also comes with a handicap. The required computation time for 3D reconstruction will be proportional to the number of images.

The final aspect for evaluation is the distribution of the viewing angles. Most of the studies in the literature do not care about this design element. However, effective distribution of the viewing angles may be very helpful in reducing the computation time

of the system. The uniform distribution of the angles can provide more information for 3D reconstruction with SFS and also lead to fewer necessary snapshots.

3.2. Design Types

There are lots of different system designs in the shape from silhouette literature. They can be classified into three main groups: multi camera systems, turn table systems, and mirror systems.

3.2.1. Multi Camera Systems

Multi-camera systems, as the name suggests, are systems which use multiple cameras. An example of a multi camera system can be seen in Figure 3.1. These kinds of systems have huge advantages and disadvantages.

Limitless design possibilities exist for these systems. Cameras can be placed as desired in these systems. Also multi-camera systems can be used for moving objects as well as stationary objects (Cheung et.al, 2005, Michoud et.al, 2006).

In contrast to these great advantages, these systems also have disadvantages. Firstly, whenever they are used for moving objects, these systems need a simultaneous trigger mechanism. Secondly, the number of snapshots depends on the fixed number of cameras. Finally, the cost of the system can be prohibitively expensive. Moreover camera calibration can be a challenging task since each camera needs to be calibrated.

Multi-camera systems can be preferable because of the freedom in design. Optimal viewing angles can be constructed using these systems at the expense of increased cost.

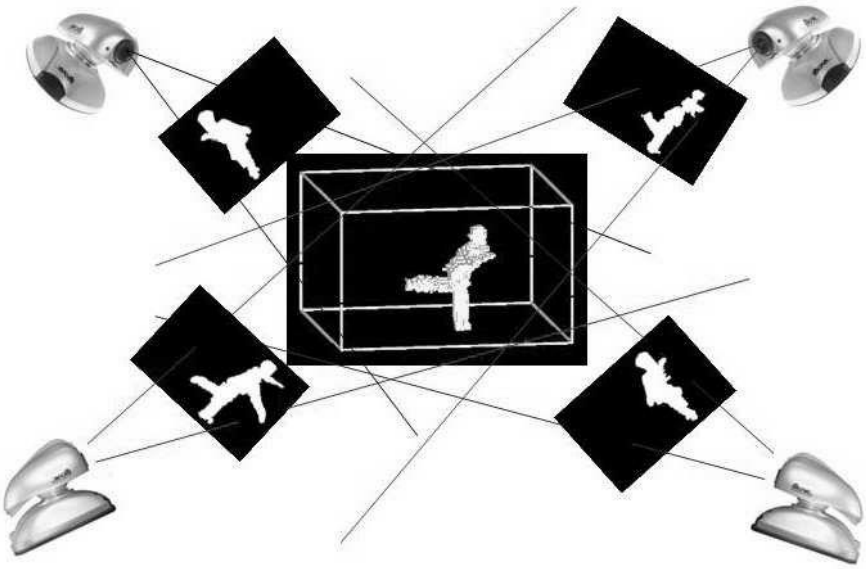

Figure 3. 1. Example of the multi camera system

3.2.2. Turn table Systems

Turn table systems consist of a camera, a platform that can turn, and a rotating mechanism. In these systems, a single camera is used. The object is rotated slowly thanks to the rotating mechanism. These systems have their own pros and cons.

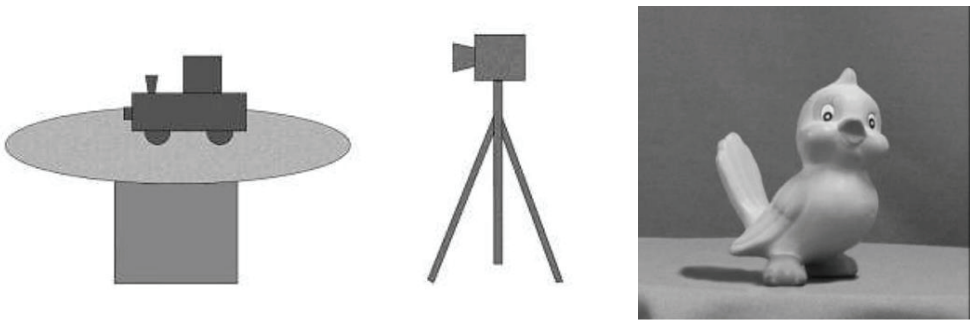

Figure 3. 2. A turn-table system (left), a picture taken using this system (right) (Olsson, 2002)

Turn table systems generally use only one camera. Therefore, cost is less compared to multi-camera systems. Also, camera calibration is not a big problem because of the use of single camera in a well-controlled environment. Another advantage of these

systems is the number of snapshots. A large number of images can be acquired leading to good reconstruction results. Equation 3.1 gives the number of snapshots. In this equation, N_s is the number of snapshots, and $R_a°$ is the rotation angle of the turn table for every steps.

$$N_s = \frac{360°}{R_a°} \tag{3.1}$$

On the other hand, there is no design variety in these systems. Since images are acquired on only one plane, information about the top and bottom of the object is limited. Also, moving objects are not suitable for these systems due to the image acquisition process. Only stationary objects can be captured. Lastly, extra materials such as step motors and rotatable platforms add load to the system budget.

Turn table systems could be preferable for a stationary objects because of the large number of snapshots, which guarantees a good reconstruction. Uniform distribution of viewing angles cannot be realized because all points of view are on the same plane. Thus, there will always be a lack of information about the object. Although they are better than multi camera systems with comparable camera count in terms of cost, they can be very slow.

3.2.3. Mirror Systems

These systems use a single camera and strategically placed mirrors to capture images of objects from different viewing angles. An example of these systems can be seen in Figure 3.3.

Mirror systems are very advantageous in terms of cost because the materials required are inexpensive, and only one camera is used. They can be used for moving or stationary objects. Unlike multi-camera systems, they do not need any trigger mechanisms for a moving objects, reducing total cost and complexity

Despite the advantages of these systems, there are also some disadvantages. The optimal system could not be designed because the distance between the real camera and the object cannot be equal to the distance between the virtual cameras and the object, even

if evenly distributed viewing angles were realized. Also, there are many other design considerations;

- Mirrors must be within the real camera's viewing angle.
- The object should not cover the mirrors.
- The reflection should be seen in the mirror by the camera.
- Primary reflections should not intersect with secondary reflections.
- Primary reflections should not cover secondary reflections.

Figure 3. 3. Example of the mirror systems (Huang et.al, 2006)

Mirror based systems are especially preferable when moving objects needs to be captured at a low budget. An optimized mirror based system is used in this thesis for real-time 3D acquisition.

3.3. Optimal System Design

Optimal system design can be described as a system with uniformly placed cameras equally distant to the center of a sphere. The cameras can be placed at the corners of the polyhedron. These systems have two important features.

- Evenly distributed viewing angles (for the case of tetrahedron).
- All cameras have an equal distance from the center where the object is located.

Systems with these features can only be constructed using multiple cameras (Figure 3.4.)

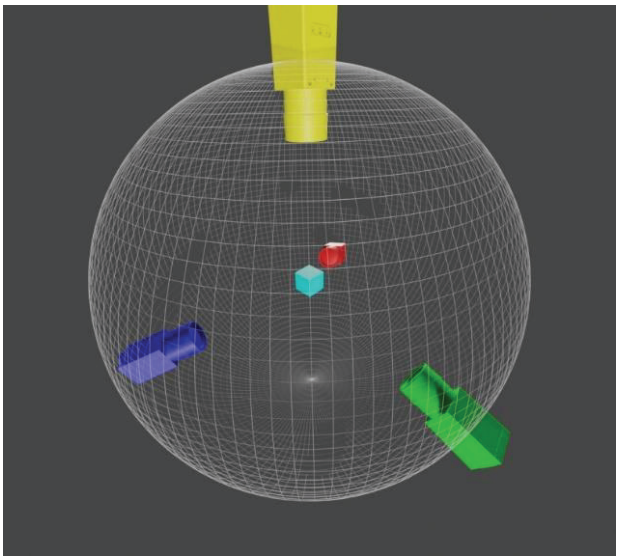

Figure 3. 4. Optimal system with multi cameras

In the mirror systems, the first feature of the optimal system can be achieved. However, the second one is impossible to accomplish because the mirror plane passes through the object when the camera and a virtual camera are equidistant from the object. Therefore, the reflection of the object cannot be seen in the mirror. In Figure 3.5, an optimal system has been tried to install with mirrors. Here, C shows a real camera while V_C represents one of the virtual cameras. They are equidistant from the object O. A mirror must be placed in the middle of the distance between V_C and C to satisfy equidistant condition and the mirror normal must be parallel to the line $|CM_c|$. In this case, it is impossible to see the object reflection because the mirror and the object intersect.

3.4. Proposed Design

When creating designs with mirrors, there are unlimited options regarding where the mirror should be placed or which direction it should be facing. In this section, a novel design that has evenly distributed viewing angles is introduced. Regular tetrahedron

corners are used while creating the design. The mirror plane is always positioned to pass through one corner of the tetrahedron.

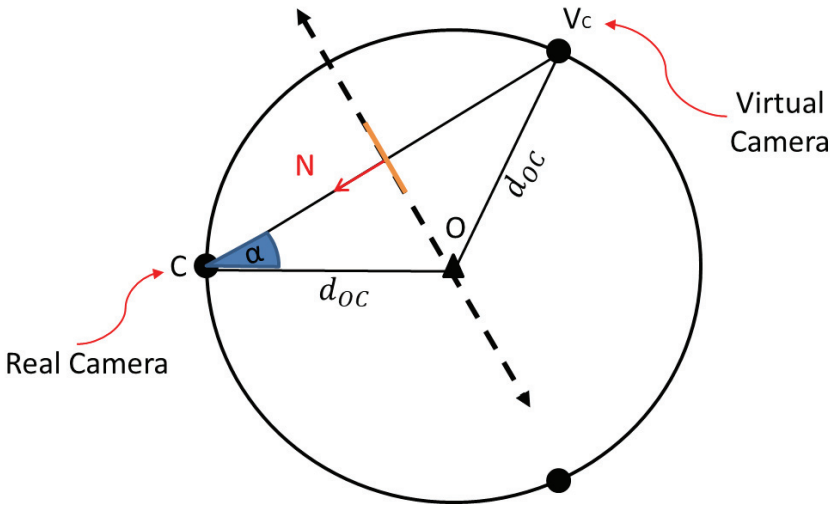

Figure 3. 5. A section of optimal system that pass through C , V_C, and O

To define location and pose of the mirror plane, a point contained by the mirror plane and a line perpendicular to the plane is required. To satisfy the viewing angle constraint, the mirror normal can be chosen as the bisector of the $\widehat{CMO}$ angle, as shown in Figure 3.6.

The user needs to determine the distance between the camera and the object. This distance which is named d_{oc} is also the distance between the corner and center of the tetrahedron. The geometric properties of the regular tetrahedron are in Appendix A. The necessary equations for the design are given below.

Let define d'_{OC} as the distance between V_c and O. So d'_{OC} become,

$$d'_{OC} = A + d_{OC} \tag{3.2}$$

where A is one of edges of tetrahedron. Let define a coefficient k

$$k = \frac{d'_{OC}}{d_{OC}} = \frac{A + d_{OC}}{d_{OC}} = 1 + \frac{A}{d_{OC}} \tag{3.3}$$

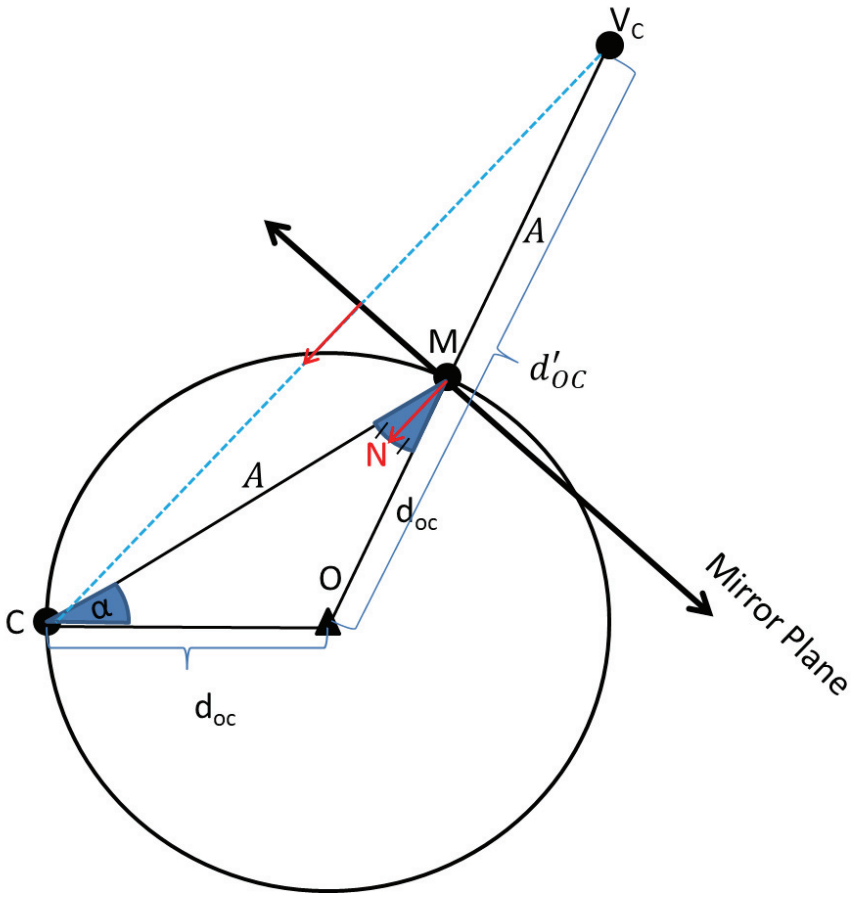

Figure 3. 6. A 2D cross-section of the proposed design.

Because of its unique geometric properties, all edges of the regular tetrahedron can be found if only one of them is known. So $A = \frac{4\sqrt{3}}{3\sqrt{2}}\, d_{OC}$. Therefore, k becomes 2.633.

3.4.1. Evaluation of the design

The novel design is created so that the mirror normal is the bisector of the $\widehat{CMO}$ angle. In this thesis, this setup with a single camera and three mirrors is used to capture evenly distributed viewing angles. The only drawback from the optimal system is the scaling observed in the mirror images. Figure 3.7 shows the novel design.

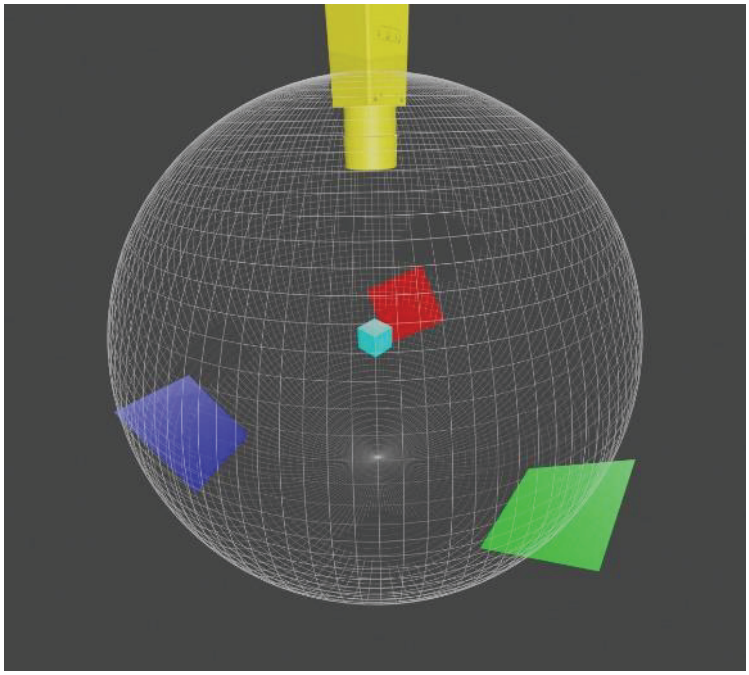

Figure 3. 7 The proposed design

CHAPTER 4

CALIBRATION

Camera and lens manufacturers specify parameters such as the pixel size of the camera sensor or the focal length of the lens. However, it is very important to calibrate the camera before using it in any application, as fabrication errors can occur. Because of this reason, camera calibration is a vital part of computer vision applications.

Camera calibration is performed to determine intrinsic parameters such as lens focal length, image center, pixel size, and skew coefficient, extrinsic parameters such as camera position and pose, and any optical aberrations.

In this thesis, multiple camera calibration algorithms are used. In this section, the basics of calibration, fisheye camera calibration, and pinhole camera calibration will be explained, respectively. Finally, it will be discussed why these algorithms are used.

4.1. Fundamentals of Camera Calibration

4.1.1. Coordinates in Calibration

Multiple coordinates are used in camera calibration algorithms. All cameras have world coordinates that cameras accept as a reference. In addition, cameras have own coordinates which named camera coordinates. These two coordinates are three-dimensional. Also, we typically use an image coordinates defined on the sensor plane where the image is formed and a discrete frame coordinates to express pixel positions. These two systems are two-dimensional coordinates. Figure 4.1 shows camera coordinates and image coordinates, Figure 4.2 shows relationship between world

coordinates and camera coordinates, and image coordinates and frame coordinates can be seen in Figure 4.3.

4.1.1.1. World Coordinates

The world coordinates are the reference for all cameras. The center of this coordinates can be inside or outside the picture, and the direction of the axes (X_w, Y_w, Z_w) may vary according to the calibration algorithm used. In this thesis, the points in the world coordinates will be shown as $\boldsymbol{P_w} = [X_w,\ Y_w,\ Z_w]^T$.

4.1.1.2. Camera Coordinates

Each camera has its own camera coordinates. The center of this coordinates is the principal point of the camera. The axes of the camera coordinates are determined as the axes of the sensor plane (X, Y) and the third axis (Z) perpendicular to them. As shown in Figure 4.1, the direction of the camera is always in the positive Z direction. In this thesis, the points in the camera coordinate system will be shown as $\boldsymbol{P} = [\mathrm{X},\ \mathrm{Y},\ \mathrm{Z}]^T$.

4.1.1.3. Image Coordinates

Image coordinates are two-dimensional. The center of the camera sensor is the center point, and its axes are parallel to the edges of the camera sensor. In this thesis, the points in the image coordinates will be shown as $\boldsymbol{p} = [\mathrm{x},\ \mathrm{y}]^T$.

4.1.1.4. Frame coordinates

The frame coordinates are formed by moving the center of the image coordinates to the upper left corner of the sensor and converting its unit to pixel counts. In this thesis, the points in the frame coordinates will be shown as $\boldsymbol{p_{px}} = [x_{px},\ y_{px}]^T$.

4.1.2. Camera Models

4.1.2.1 Pinhole Camera Model

The pinhole camera model, which is the simplest camera model, describes the mathematical relationship for the projection of points in 3D-space onto an image plane. It can be seen in the Figure 4.1.

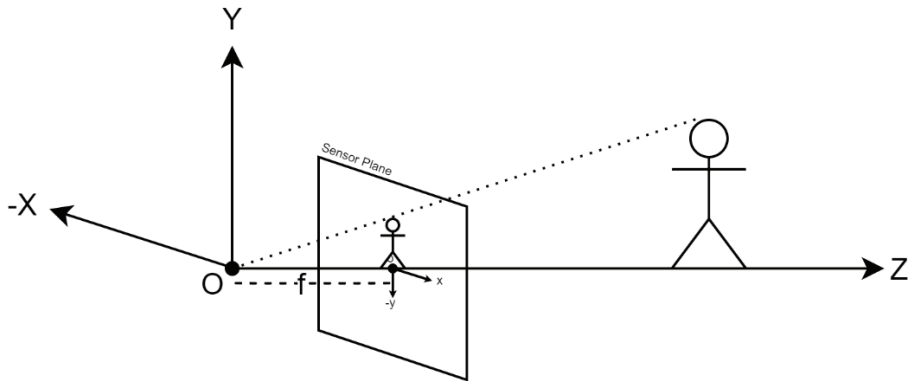

Figure 4. 1. Pinhole Camera Model

In applications that work with multiple cameras, each camera has its own coordinates. In contrast, the world coordinates are unique. Any point in the world coordinates ($\boldsymbol{P_w}$) can be described by the camera's coordinates. Figure 4.2 demonstrates the relationship between them. For this process, the rotation matrix and translation vector of each camera are needed. Equations 4.1 and 4.2 show the relationship between the

camera's coordinates and the world coordinates. In these equations, **R** represents the rotation matrix and **T** represents the translation vector.

$$P = R.P_w + T \tag{4.1}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} . \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix} \tag{4.2}$$

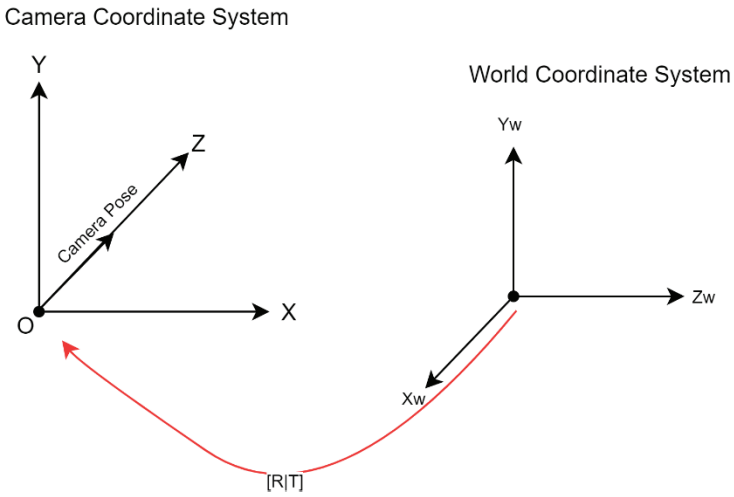

Figure 4. 2. Projection from world coordinates to camera coordinates

The rotation matrix and the translation vector are called extrinsic parameters, which can be combined as equation 4.3.

$$M_{ext} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & T_X \\ r_{2,1} & r_{2,2} & r_{2,3} & T_Y \\ r_{3,1} & r_{3,2} & r_{3,3} & T_Z \end{bmatrix} \tag{4.3}$$

In this case, $\boldsymbol{P_w}$ is arranged in homogeneous coordinates as in equation 4.4.

$$P_w = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.4}$$

After finding the equivalent of a point in the camera coordinates, projecting it to the image coordinates is a straightforward operation. Equation 4.5 shows this calculation.

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = f. \begin{bmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{bmatrix} \tag{4.5}$$

where f stands for the focal length of the camera.

To switch from the image coordinates to the frame coordinates, the center point of the image coordinates must be moved to the upper left corner of the sensor plane, as seen in Figure 4.3, and the units should be converted to pixels. Equation 4.6 shows this projection. Here, O_x and O_y represent the center point of the image in the frame coordinates, and s_x and s_y represent the side lengths of the pixels along the x-axis and y-axis, respectively.

$$p_{px} = \begin{bmatrix} x_{px} \\ y_{px} \end{bmatrix} = \begin{bmatrix} O_{x_{px}} \\ O_{y_{px}} \end{bmatrix} - \begin{bmatrix} x/s_x \\ y/s_y \end{bmatrix} \tag{4.6}$$

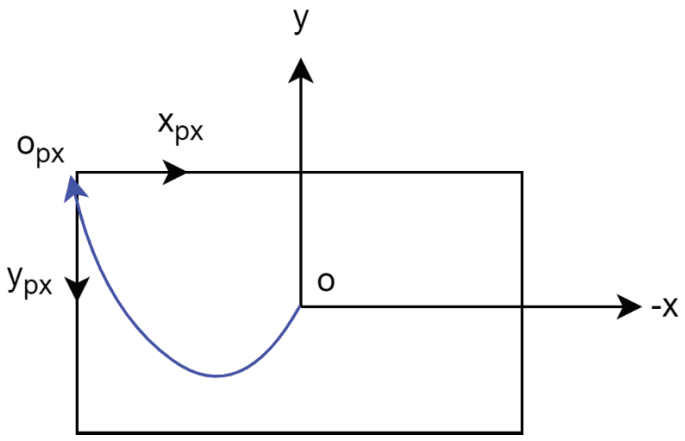

Figure 4. 3. Projection between image coordinate system to frame coordinate system

In pinhole camera model f, s_x, s_y, O_x, O_y, and skew coefficient are classified as intrinsic camera parameters. They can be written as in Equation 4.7. Here, the skew coefficient, denoted s, is usually assumed to be equal to 1.

$$M_{int} = \begin{bmatrix} \frac{f}{s_x} & 0 & 0 \\ s & \frac{f}{s_y} & 0 \\ O_x & O_y & 1 \end{bmatrix} \tag{4.7}$$

4.1.2.2 Omnidirectional Camera Model

The omnidirectional camera model was developed to mathematically express fisheye and omnidirectional lenses, as well as catadioptric systems because pinhole camera model is not sufficient to represent these systems. The extrinsic parameters of the omnidirectional camera model are the same as the pinhole camera model (Equation 4.3). But the intrinsic parameters are completely different.

Let $\boldsymbol{p_{px}} = [\, u_{px}\,, v_{px}\,]^T$ be a point in the frame coordinates, and its correspondence point in the image coordinates is $\boldsymbol{p} = [\, u\,, v\,]^T$. These two points are related by an affine transformation as shown in the equations 4.8 and 4.9. The transformation matrix A is for small misalignment between the sensor and lens.

$$p_{px} = A.\,p + Oc \tag{4.8}$$

$$\begin{bmatrix} u_{px} \\ v_{px} \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} . \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} O_x \\ O_y \end{bmatrix} \tag{4.9}$$

At this point, imaging function g, which is the relationship between the point $\boldsymbol{p_{px}}$ in the frame coordinates and the point **P** in the camera coordinates, needs to be introduced.

$$\lambda g(m) = \lambda\big(u, v, f(u,v)\big)' = \lambda\big(u, v, f(\rho)\big)' \tag{4.10}$$

where λ is positive real number.

The function f depends on lens, catadioptric system and proposed projection models. For example, Scaramuzza et.al uses the Taylor series as shown in equation 4.11 where a_0, a_1, a_2 ... are coefficient of Taylor series and $\rho = \sqrt{u^2 + v^2}$ (Scaramuzza et.al, 2006-b).

$$f = a_0 + a_1\rho + a_2\rho^2 + a_3\rho^3 + \cdots + a_N\rho^N \tag{4.11}$$

4.1.3 Camera Calibration Methods

4.1.3.1 Pinhole Camera Calibration

Pinhole camera calibration algorithms are used to estimate the 17 (9 for rotation, 3 for translation, and 5 for intrinsic parameters) unknowns in the pinhole camera model. Camera calibration algorithms can be classified according to the calibration object they use (Urban et.al, 2015).

- Self-Calibration (there is no calibration object)
- Non-planar object calibration
- Planar object calibration

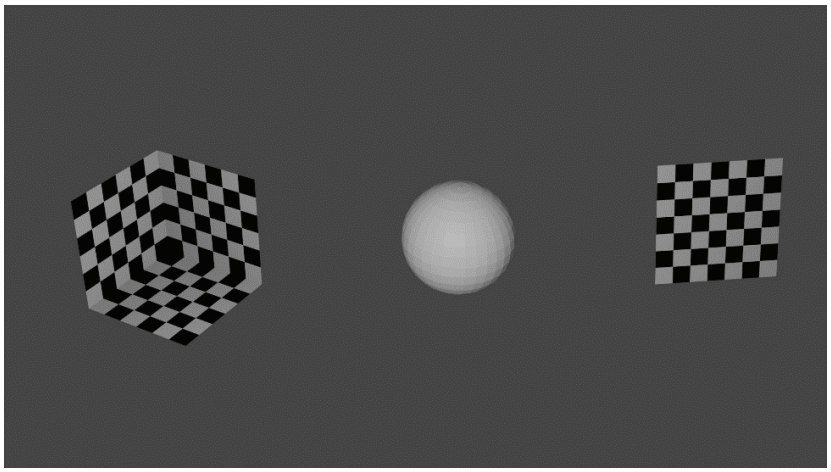

Figure 4. 4. Multi-planar (left), non-planar (middle) and planar (right) calibration objects

Self-calibration algorithms try to calibrate the camera by taking advantage of camera movements and features such as focus and zoom, where the camera can change its intrinsic parameters (Hemayed, 2003). These types of techniques are less robust than others (Urban et.al, 2015). For this reason, they are not preferred in this thesis.

For the second type of methods, the non-planar object can be a sphere (Agrawal et.al, 2003-a, 2003-b) or an object to which multiple planar planes are joined (Dawson-Howe et.al, 1994). In studies with spheres, only the radius of the sphere should be known, while in others, all 3D information of the object is needed.

Finally, calibrations with a planar object are the most reliable and easiest to implement. Many examples can be given for this type of method, but widely used one is Zhang's calibration algorithm (Zhang et.al, 2000).

In this thesis, Zhang's calibration algorithm as an example of calibration with a planar object and Guan's sphere calibration algorithm (Guan et.al, 2015) as an example of calibration with a non-planar object will be examined.

Calibration with a Planar Object

Zhang's calibration algorithm (Zhang et.al, 2000) uses a simple planar object with a checkerboard pattern on it, as shown on the right side of Figure 4.4. It is a very flexible and powerful technique because any planar surface with a checkerboard pattern can be used as a calibration object. The camera captures the planar pattern in a few different orientations to use the calibration technique. Either the camera or the planar pattern can be moved by hand.

Traditional calibration procedures usually use precisely constructed 3D calibration objects in order to calibrate the camera. Zhang's method does not need any information about 3D scene or objects. Only the 2D metric information on the planar object is sufficient for calibration. In typical implementations, the user is asked the length of the squares' edges and asked to select the checkerboard corners

The calibration algorithm assumes the checkerboard is on Z = 0 in the world coordinates, so pinhole camera model can be used as in 4.12 where $\boldsymbol{M_{ext}} = [\boldsymbol{r1} \quad \boldsymbol{r2} \quad \boldsymbol{r3} \quad \boldsymbol{t}]$.

$$\omega \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_{int}.M_{ext} \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \tag{4.12}$$

$$\omega \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_{int}.[r1 \quad r2 \quad t] \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Here, $\boldsymbol{r1}, \boldsymbol{r2}$ and $\boldsymbol{r3}$ are column vectors of rotation matrix $\boldsymbol{R}$, and $\boldsymbol{t}$ is translation vector. So $\omega \boldsymbol{p_{px}} = \boldsymbol{H}.\boldsymbol{P_w}$ with $\boldsymbol{H} = \boldsymbol{M_{int}}.[\boldsymbol{r1} \quad \boldsymbol{r2} \quad \boldsymbol{t}]$. The 3 x 3 matrix H is defined up to a scale factor. Let's denote it by $\boldsymbol{H} = [\boldsymbol{h_1} \quad \boldsymbol{h_2} \quad \boldsymbol{h_3}]$, so the equation becomes,

$$[h_1 \quad h_2 \quad h_3] = \lambda M_{int}.[r1 \quad r2 \quad t] \tag{4.13}$$

where λ is an arbitrary scaler. Using the fact that $\boldsymbol{r1}$ and $\boldsymbol{r2}$ are orthonormal

$$h_1^T M_{int}^{-T} M_{int}^{-1} h_2 = 0 \tag{4.14}$$

$$h_1^T M_{int}^{-T} M_{int}^{-1} h_1 = h_2^T M_{int}^{-T} M_{int}^{-1} h_2 \tag{4.15}$$

As a result of the closed form solution starting with $\boldsymbol{B} = \boldsymbol{M_{int}^{-T}} \boldsymbol{M_{int}^{-1}}$, the algorithm can estimate intrinsic parameters.The algorithm can be summarized as follows:

- Print the checkerboard pattern and paste it on a flat surface.
- Take several pictures of the planar object in various orientations.
- Detect the feature points in the images.
- Guess all the intrinsic parameters and extrinsic parameters using the closed-form solution.
- Solve the linear least-squares problem to estimate the radial distortion coefficients.
- Minimize all parameters to perfect them.

As discussed above Zhang's algorithm is a powerful and robust algorithm. The algorithm only needs several pictures of the checkerboard pattern in different orientations. It is easier to use and flexible compared to classical algorithms using fragile calibration objects.

Calibration with Non-Planar Object

In this subsection, the calibration objects can be divided into two classes, multi-planar and non-planar. The multi-planar objects can be very complex objects, as shown in Figure 4.4 (left). The geometrical properties, such as the angle of intersection of planes, are generally known. It is quite difficult to construct and maintain such objects. However, the non-planar objects are usually simple, easy to find, and inexpensive. They can be a

simple sphere whose radius is known only, as in Figure 4.4 (middle). Calibration algorithms with spheres are much easier to apply and are more flexible than classical algorithms.

Camera calibration algorithms with spheres have been attractive interest in calibration literature, and there are many new studies in it. Two of these studies were done by Agrawal et al. (2003-a, 2003-b) in 2003. They published two methods which are named a dual-space approach and a semi-definite programming approach. Their camera calibration algorithm uses the occluding contours of spheres. The camera must examine a sphere from three or more angles in order to calculate calibration parameters. The solution is precise and doesn't require any initialization. However, the accuracy of calibration results is heavily influenced by the quality of boundary fitting and ellipse identification. Zhang et.al (2005) presented two different methods named as scalar and orthogonal approaches in a single article. Scalar approach, which is a linear solution, is derived from Agrawal's semi-definite programming approach. However, this approach is very sensitive to noise. A second approach was presented to solve this problem, in which the orthogonal calibration relationship is derived by treating any two spheres as a surface of revolution. The orthogonal approach overcomes a major limitation of Agrawal's systems, specifically the quality of conic detection, which has a significant impact on the accuracy of calibration results. Zhang et.al. (2007) published Camera Calibration from Images of Spheres in 2007. In this work, they made improvements to their previous work by defining extra parameters. Lu et.al (2010) published another sphere calibration algorithm in 2010. Based on conic extraction, they increased the accuracy of camera calibration from sphere images. They adopted a conic orientation assumption so that the conic's primary axis goes through the image's optical center. As a result, the orientation can be calculated uniquely and correctly. Wong et. al. (2010) proposed a calibration algorithm that has no assumptions on the camera intrinsic parameters in 2010. Since no iterative optimization is involved, it is computationally efficient. Guan et. al. (2015) published an algorithm in 2015 that only calculates extrinsic parameters of the calibration. They assume that the intrinsic parameters are already known. They provide a simple and precise approach for calculating the 3D position of the sphere's center in relation to the local camera coordinate system. Also, they propose using orthogonal procrustes analysis to pairwise estimate the camera's relative extrinsic parameters. Also in 2015, Penne et. al. (2015) developed a new method for calculating focal length using a single image of a

single ball. Penne et. al. (2019) reveal a new method for accurately localizing the center of a sphere with a known radius relative to the camera reference system using a single calibrated image of the sphere. Using the center and the major size of the ellipse, and the intrinsic of the pinhole camera, they derived a formula for the position of a sphere center with a known radius. In this thesis, extrinsic camera calibration of Guan will be examined together with robust sphere localization algorithm of Penne.

In this thesis, Guan's extrinsic camera calibration algorithm is employed. In this algorithm, it is assumed that the intrinsic parameters are known. They demonstrate an accurate and easy method for estimating the 3D information of the sphere. However, in this thesis, position information is calculated using Penne's algorithm because it is more precise and robust than Guan's algorithm. Then Guan's orthonormal procrustes analysis is implemented to calculate extrinsic parameters. Finally, an optimization algorithm is employed to refine the extrinsic parameters for the camera and virtual cameras.

In order to estimate the properties of the spherical object, we need to find the ellipse center (M), semi-major axis (k), and semi-minor axis (l) of the ellipse from the image. Let Sm represent the distance between image center (C) and ellipse center. Then φ and θ can be calculated with these formulas. Figure 4.5 shows Sm, $\boldsymbol{C}$, $\boldsymbol{M}$, k, l, φ, and θ.

$$\varphi + \theta = Arctan\left(\frac{Sm + k}{f}\right)$$
$$\varphi - \theta = Arctan\,(\frac{Sm - k}{f}) \quad \textbf{(4.16)}$$

From the angle θ, the distance $|OP|$ can be derived where O is the pinhole center.

$$|OP| = r\,\frac{\sqrt{tan^2(\theta) + 1}}{tan(\theta)} \quad \textbf{(4.17)}$$

where r is the radius of the sphere. From the angle φ, the sphere center $\boldsymbol{Q}$ can be extracted in pixel unit.

$$Q = C + \frac{f.tan(\varphi)}{Sm}(M - C) \quad \textbf{(4.18)}$$

After these calculations, the center of the sphere $\boldsymbol{P}$ can be acquired with $\boldsymbol{C} = (uo, v0)'$, $\boldsymbol{Q} = (u1, v1)'$ and $|OQ| = \sqrt{(u1 - u0)^2 + (v1 - v0)^2 + f^2}$.

$$P = \frac{|OP|}{|OQ|}(u1 - u0, v1 - v0, f)' \tag{4.19}$$

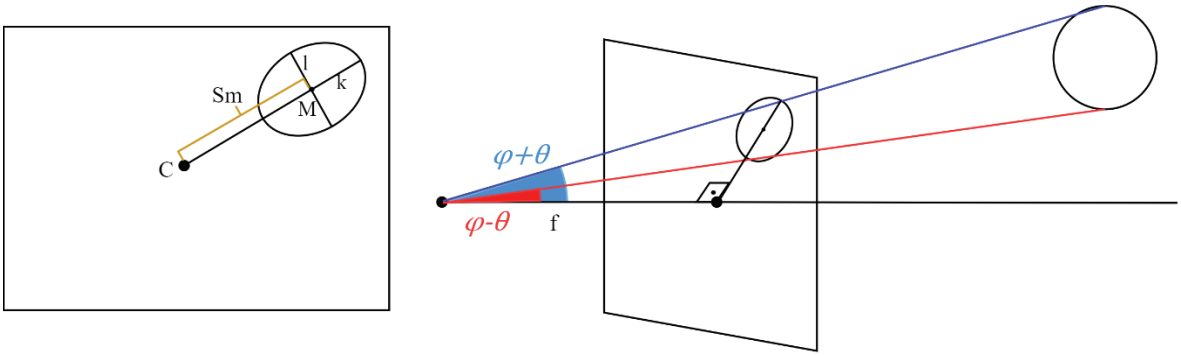

Figure 4. 5 Sphere projection on an image plane (right) and its 3D geometry (left)

After estimating, the center of the sphere, Guan examines two different ways to calculate the extrinsic parameters of the camera. The first one involves Rank-4 factorization, and the second one utilizes orthogonal procrustes. The author stated that orthogonal procrustes gave better results than Rank-4 factorization. For this reason, in this thesis, orthogonal procrustes method is preferred.

Let's define $\boldsymbol{r_i^k}$ where $i = 0,1,2, \ldots N$ and $k = 0,1,2, \ldots L$. N is the number of taken pictures, L equals the camera number.

$$r_i^k = \begin{bmatrix} P_1^k & P_2^k & \cdots & P_N^k \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{4.20}$$

Then, mean of the $\boldsymbol{r_i^k}$ denoted as $\overline{\boldsymbol{r^k}}$ is needed to estimate the translation and rotation (Guan et.al, 2015).

$$\overline{r^k} = \frac{1}{N}\sum_{i=1}^{N} r_i^k \tag{4.21}$$

Let's $\boldsymbol{H^1}$ be a matrix with columns $\boldsymbol{r_i^1} - \overline{\boldsymbol{r^1}}, i = 0,1 \ldots N$ and $\boldsymbol{H^k} = \boldsymbol{r_i^k} - \overline{\boldsymbol{r^k}}$.

$$H^k = R^k H^1$$
$$H^1(H^k)^T = U_k S_k V_k \quad \textbf{(4.22)}$$

To decompose $\boldsymbol{H^1(H^k)^T}$, singular value decomposition is used, then **R** matrixes can be calculated with this equation.

$$R^k = V_k {U_k}^T \quad \textbf{(4.23)}$$

Once, **R** matrixes are obtained, translation vectors are,

$$T = \overline{r^k} - R^k \overline{r^1} \quad \textbf{(4.24)}$$

After, all extrinsic parameters are obtained. An optimization algorithm is used to minimize the total reprojection error.

In the above algorithm, since the rotation matrices are calculated according to the first camera, the world coordinate system and the first camera coordinate system are the same.

4.1.3.2 Fisheye Camera Calibration

Fisheye camera calibration was introduced because pinhole camera calibration was considered inadequate for some cameras. There is no standard camera calibration model in this area. Kannala et.al (2008) uses the generalized camera model, while Scaramuzza et.al (2006-b) uses the omnidirectional camera model. There is a slight difference between these models. This difference causes different performance outcome with different lenses.

Kannala et al (2008) claim that pinhole camera calibration is insufficient for fisheye camera calibration. They criticize the use of a preprocessing step for the conversion of original fisheye image to a perspective image for pinhole calibration. As a result, they propose a new camera model that will work with both omnidirectional and conventional cameras in 2006. After that, Scaramuzza et. al. (2006-b) developed a

calibration algorithm with a slightly different omnidirectional camera model. They use Taylor series expansion whose coefficients are estimated with a two-step least square linear minimization. They claim that their algorithm can be used in all omnidirectional cameras, both dioptric and catadioptric. Scaramuzza et.al (2006-a) also published the camera calibration algorithm as a toolbox for MATLAB. In this toolbox, in addition to the linear minimization problem, they used a maximum likelihood based non-linear refinement algorithm. Finally, they reduced the number of intrinsic parameters based on their observations. Urban et. al. (2015) modified Scaramuzza's linear refinement algorithm and developed a more robust algorithm. They also reduced the number of steps required for the original calibration algorithm. Unlike others, Mei et. al. (2007) advocate that polynomial approximations lead to impractical calibration methods. Therefore, they proposed a unified projection model which is similar to the pinhole camera model.

In this thesis, Scaramuzza's algorithm is utilized because it is a very flexible and robust algorithm, and it has been shared as a MATLAB toolbox. Scaramuzza's camera model projection equation is;

$$p_i = \lambda \begin{bmatrix} u_i \\ v_i \\ a_0 + \cdots + a_N \rho_i^N \end{bmatrix} = M_{ext} P_w \tag{4.25}$$

In this algorithm, more than two images of the checkerboard pattern in different orientations are needed to find the calibration parameters. Then the vertices of the checkerboard are determined, and it is assumed that the planar object is located on the $Z = 0$ axis. Thus, 3D information of each point in the pattern is obtained. Since the pattern is on the Z = 0 axis, the equation can be arranged as follows.

$$\lambda \begin{bmatrix} u_i \\ v_i \\ a_0 + \cdots + a_N \rho_i^N \end{bmatrix} = [r1 \quad r2 \quad t] \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{4.26}$$

If both sides of the above equation are multiplied by p_i, the following three equations can be found.

$$v(r_{31}X + r_{32}Y + t3) - f(\rho)(r_{21}X + r_{22}Y + t2) = 0 \tag{4.27}$$

$$f(\rho)(r_{11}X + r_{12}Y + t1) - u(r_{31}X + r_{32}Y + t3) = 0 \tag{4.28}$$

$$u(r_{21}X + r_{22}Y + t2) - v(r_{11}X + r_{12}Y + t1) = 0 \quad \textbf{(4.29)}$$

The unknowns in the equation 4.29 $r11, r12, r21, r22, t1, t2$. If these equations are used in an L point calibration model,

$$MH = 0 \quad \textbf{(4.30)}$$

where,

$$H = [r11, r12, r21, r22, t1, t2]$$

$$M = \begin{bmatrix} -v_1X_1 & -v_1Y_1 & u_1X_1 & u_1Y_1 & -v_1 & u_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -v_LX_L & -v_LY_L & u_LX_L & u_LY_L & -v_L & u_L \end{bmatrix} \quad \textbf{(4.31)}$$

This equation is solved with SVD, and the unknown parameters mentioned above are found. Because of the orthonormality, constraint $r31$ and $r32$ can also be calculated uniquely. Thus, all external parameters except $t3$ are found. Then the estimated parameters are placed in the equations 4.27 - 4.29, and the remaining t3 and intrinsic parameters are found.

This procedure is repeated until the average value of the reprojection error of all calibration points decreases below a certain range. The N value, which determines the degree of projection polynomial ($a_0 + \cdots + a_N\rho_i^N$), is initially set to $N = 2$ and incremented by one at each iteration.

4 .2 Implementation of Calibration on the System

In this subsection, a camera with a fisheye adaptor and three plane mirrors that are positioned at equal distances is calibrated, which is shown in Figure 4.6.

In the literature, there are systems constructed with many cameras and mirrors. Most of these studies have found different solutions to the calibration problem. Forbest et.al (2004,2006) need second reflection of an object to find f and principal point. They only calibrate the real camera and calculate the other parameters geometrically. In the

study Gluckman et.al (2001), normal of the mirrors and distances from the mirrors to the camera are needed. In Kumar et.al (2008) and Takahashi et.al (2012), a camera cluster with non-overlapping fields of view was calibrated. Firstly, they calibrated the mirror camera and then established its relationship with the real camera. In the study of Rodrigues et.al (2010), the camera is assumed to make a certain movement. The study of Feng et.al (2018) does not require any prior knowledge, but the results are not sensitive enough. Finally, Yin et.al (2019) does not require any prerequisites. However, a planar object is used as the calibration object, and their system can capture the object in a single frame.

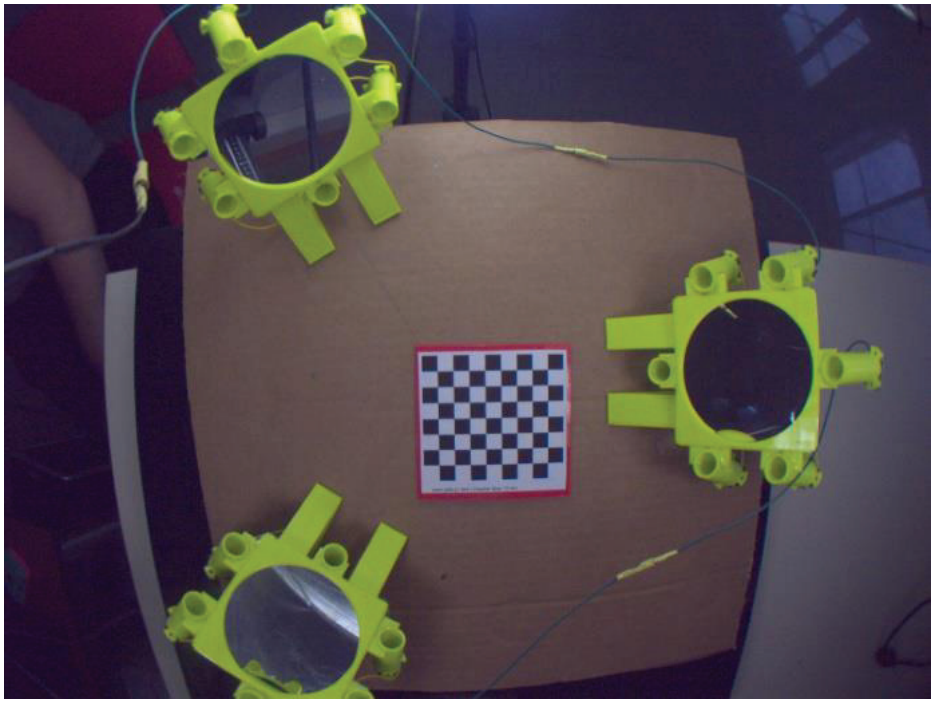

Figure 4. 6 An image of the system taken form real camera

The calibration algorithms with a planar object, such as Scaramuzza et.al, (2006-b) or Urban et.al, (2015), are very flexible and stable algorithms. However, these algorithms also have disadvantages. One of the disadvantages is that the cameras do not have a common field of view. Another problem is that even though they have a common field of view, not all cameras can see the calibration object at the same time. This is because the calibration object is single-sided. Our setup also suffers from this problem since the real camera and virtual cameras are evenly distributed. While three virtual

cameras see the calibration object, the real camera stays behind the object. For these reasons, calibration with a planar object is not sufficient to calibrate our design.

For the reasons stated above, spherical calibration object is used which can be seen by all cameras simultaneously. Sphere calibration algorithms often make use of the pinhole camera model. For this reason, the fisheye images obtained should be projected onto the pinhole camera model.

For this purpose, Scaramuzza's omnidirectional camera calibration algorithm was employed. This algorithm can be easily accessed via MATLAB. With the intrinsic parameters obtained as a result of the calibration, the center shift problem between the lens and the sensor is solved, and the image can be reprojected using pinhole model (Figure 4.7).

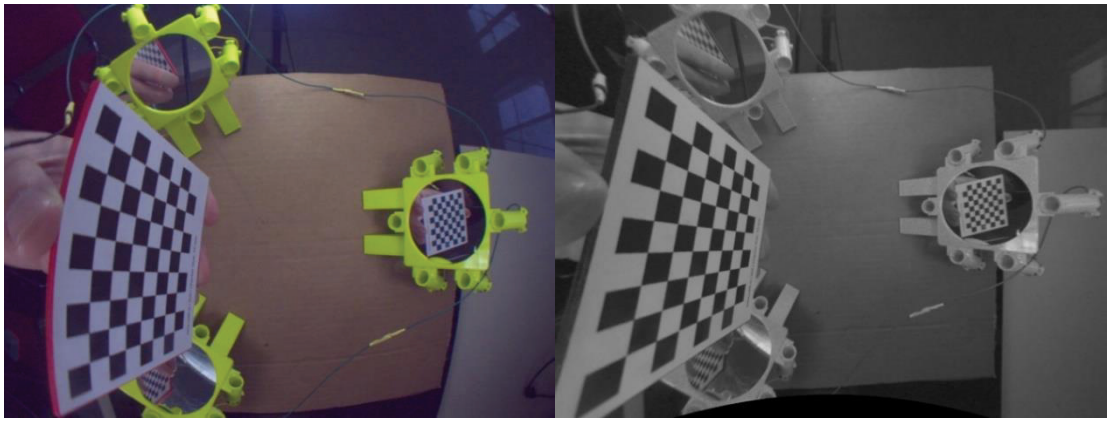

Figure 4. 7. A fisheye image on the left and an undistorted image on the right.

In our implementation, backward mapping is done by using inverse Taylor series coefficients. In addition, faster results can be obtained by utilizing CUDA, since this projection can be parallelized. Finally, the ratio of focal length to the horizontal axis of the image sensor can be represented by the coefficient ks in the backward mapping algorithm. Figure 4.8 demonstrates the cases for different ks values.

Although calibration of both intrinsic and extrinsic parameters is possible using the spherical object, our experimental results showed signs of unreliability (e. g. especially when the spherical object is close to the image center), therefore we chose to

use Zhang’s method for the calibration of intrinsic parameter on the perspective images reprojected from fisheye images.

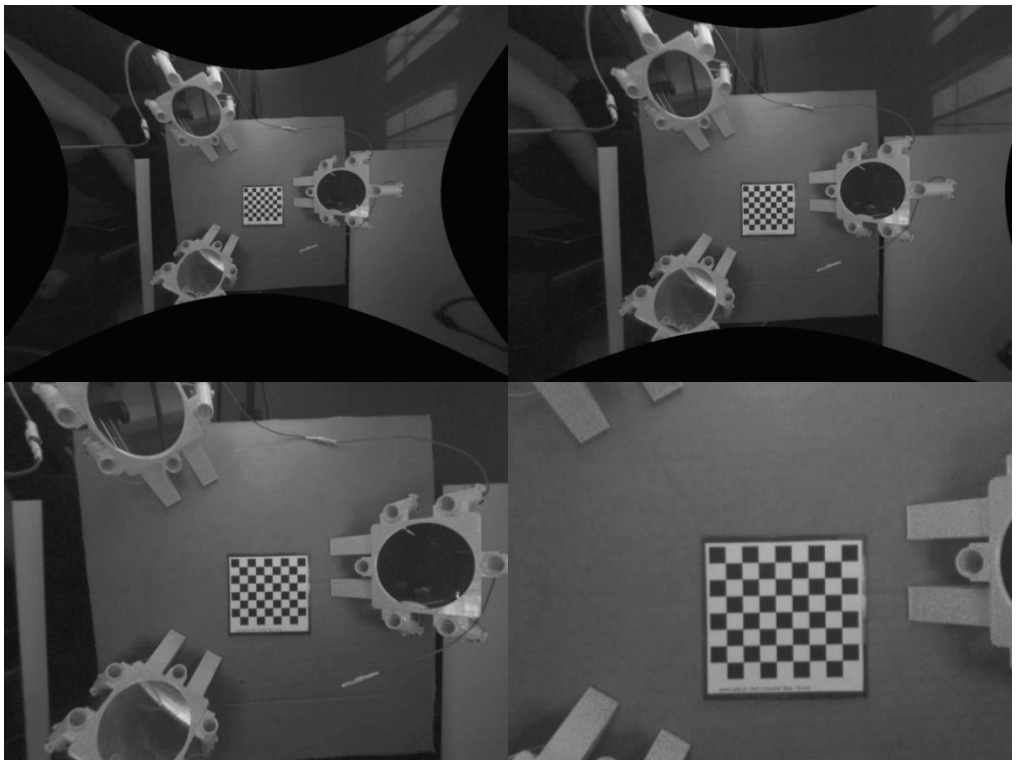

Figure 4. 8. Undistortion of images with different ks values

Guan's extrinsic camera calibration algorithm (Guan et.al, 2015), which was developed with Penne's sphere localization algorithm (Penne et.al, 2019), was used to find the extrinsic parameters. The pictures in Figure 4.9 are used to localize sphere centers. To calculate extrinsic parameters, centers of spheres are used as inputs of Guan’s algorithm. This completes the calibration of the camera.

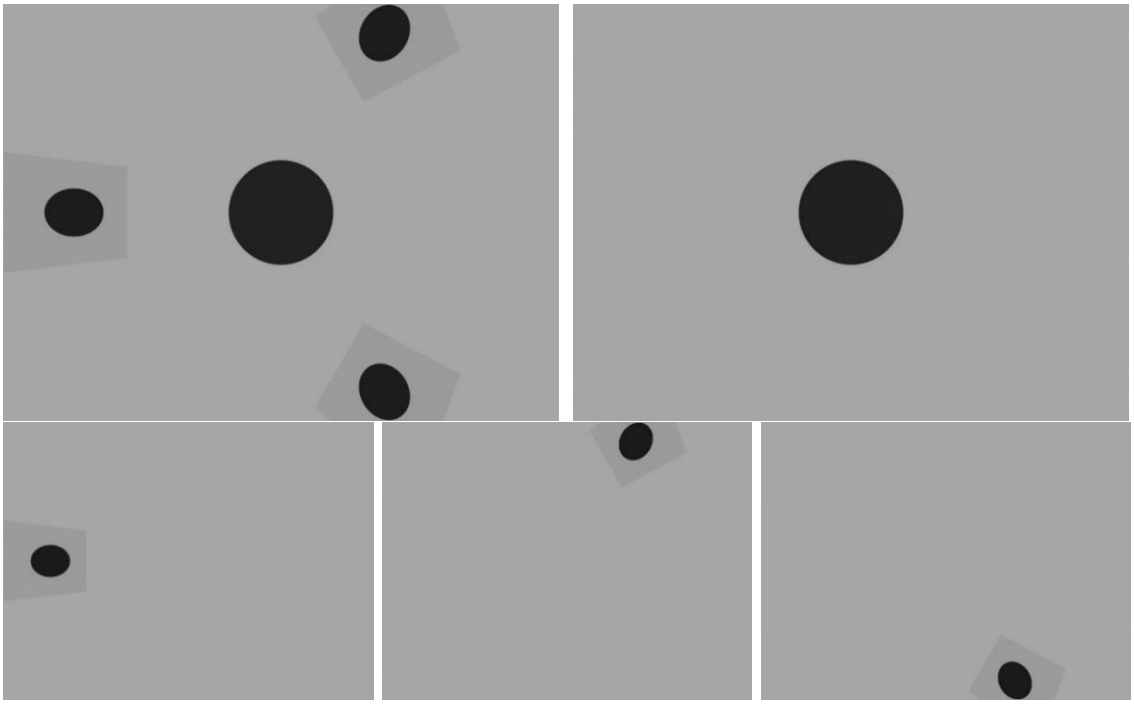

Figure 4. 9. Sample calibration image and its segmented parts.

CHAPTER 5

SHAPE FROM SILHOUETTE

3D reconstruction is one of the fastest growing fields in computer vision. It can be used in many fields, from the gaming industry to medical science. The demand for technologies such as augmented reality and virtual reality motivates research for efficient techniques of capturing 3D data.

There are many types of 3D reconstruction algorithms, such as shape from stereo-vision or shape from texture, in the literature. Shape from silhoueete (SFS) is one of the efficient and robust 3D reconstruction methods when compared to others, and it is also the simplest method of all. It takes advantage of the silhouettes of an object captured by image taken in different orientations. Also, it can provide preliminary solutions that can be refined by further applications of other shape form X methods.

Despite its positive properties, SFS has some drawbacks. It generally produces low-quality results and cannot fully reconstruct non-convex objects. SFS can be computationally expensive when too many silhouettes with high resolution are used.

SFS methods are generally divided into two groups, which are volume-based methods and surface-based methods. Volume-based methods are easy to implement but computationally expensive. They are suitable for parallel computing. Surface-based approaches are not easy to use, but they produce results with more details.

This section includes a brief introduction of the visual hull. Then, some studies in the literature are discused. Finally, utilized methods are explained.

5.1 Visual Hull

Visual hulls are approximations of objects' shape obtained from visual silhouettes (Franco el.al, 2003). Each silhouette forms a cone with the center of the camera. Visual

hull is computed as the intersection of the visual cones formed by the silhouettes if these cones are projected to infinity (Haro, 2012).

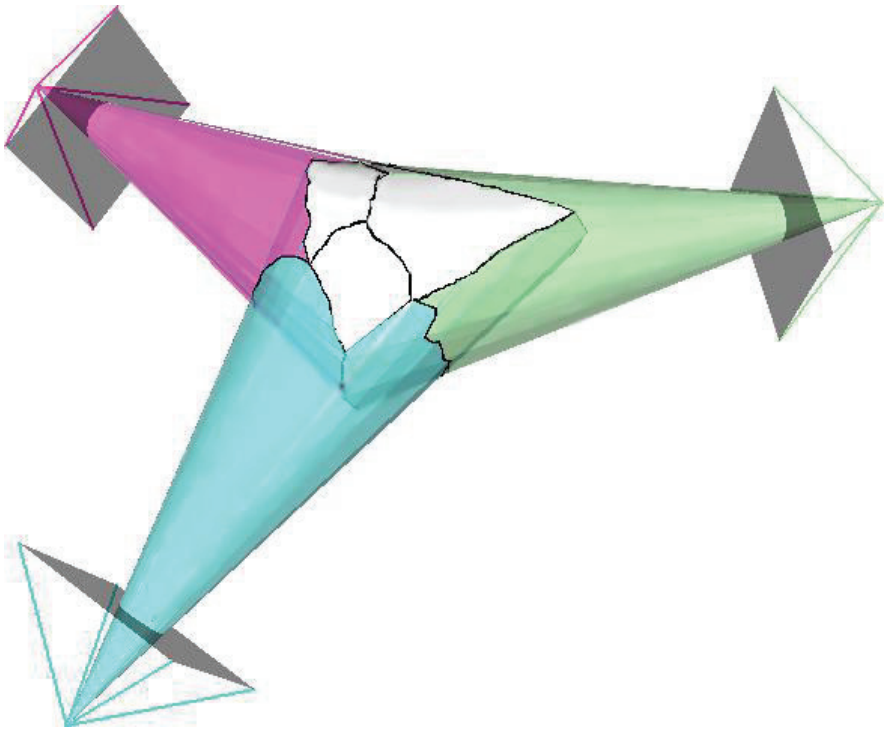

Figure 5. 1. An example of a visual hull

A convex object (e.g. a cube) can be fully represented by its silhouettes when sufficient number of viewpoints are used. However, a non-convex object such as a vase can never be the same as its visual hull. Therefore, the visual hull can be defined as the largest object that is consistent with a series of silhouettes (Forbes,2007).

5.2 Related Works

SFS studies in the literature are generally divided mainly into 2 groups;

- Volume based SFS
- Surface based SFS

Volume based methods are simpler than others. The basis of these methods is volume pixels called voxels (Cheung et.al, 2005). Voxels are usually obtained by dividing 3D space into equal parts, and they are labeled by binary. The purpose of labeling is to separate voxels that are inside or outside of the visual hull (Siudak at.al,2014). Volume-based methods can be easily parallelized, allowing lower computation times (Kolev et.al, 2012), when implemented on parallel architectures.

Volume-based methods are not entirely good. One of the problems is determining the optimal voxel size. If the voxels are too large, the reconstruction quality will be poor. If they are too small, computation time increases dramatically. Also, the results are not photo-realistic (Kolev et.al, 2012). The study of Kuzu et.al (1983) can be given as an example of volume based methods. The authors introduced voting based voxel carving, using a simple octree structure.

Octree structures are a way to iteratively compute the visual hull by reducing the size of the voxels. The purpose of this algorithm is to reduce computational time (Szeliski, 1993). Figure 5.2 shows the results of iteration steps.

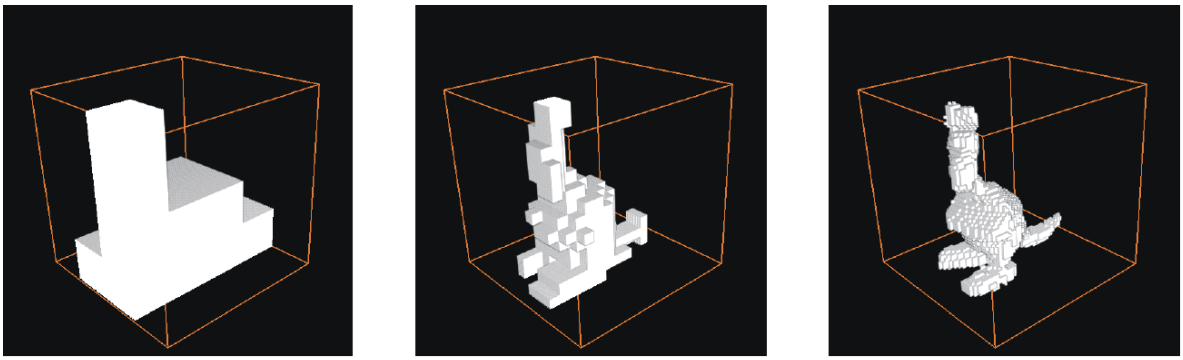

Figure 5. 2. Results of iteration steps in algorithm (Kuzu et.al, 1983)

Surface-based SFS algorithms calculate the approximated surface of the visual hull using information about the silhouette's contours. The surfaces are formed from intersections of the side surfaces of visual cones (Butime et.al, 2006). In comparison to the prior method, this one generates a photo-realistic visible hull while requiring less

memory. However, the intersections in 3D space are highly sensitive to numerical errors, especially for complex objects (Yous et.al, 2007).

Some studies in the literature calculate intersection points in 3D space (Franco et.al, 2003,2009, Yous et.al,2007), while others calculate them in 2D space (Matusik et.al, 2000,2002). Franko et.al (2008) propose a more general definition of the polyhedral visual hull. They use an algorithm to detect outlines at the boundary of the discrete silhouette and non-silhouette pixel regions. They claim that the intersection computations are unstable because of finite machine precision. To overcome this problem, they use arbitrary-precision arithmetic. Matusik et. al. (2002) take inspiration from image-based rendering. Instead of calculating a 3D object, they calculate a 2D image for each new viewpoint. This makes their algorithm very fast.

There are also studies (Tarini et.al, 2002 and Ladicos et.al, 2008) that do not belong to either of the above two groups. Tarani et.al. (2002) publish a method that utilizes both volume and surface-based features. The object space is split into a 3D uniform grid whose resolution is selected by the user. The grid is made of 3 sets of rays. Rays are aligned with the silhouette and store points of entry and exit into the volume. Each silhouette cone can be converted to the marching intersections data structure. These data are projected onto 3D space and merged. Algorithm of Ladikos et.al. (2008) also utilizes both volume and surface-based features. They compute the bounding boxes of voxel projection for all images and store them in lookup tables. Then they use those tables while calculating visual hull.

Finally, there are some SFS algorithms (Forbes et. al, 2004, 2006, Hu et.al, 2005,2009) that work with mirrors which is also one of the main subjects of this thesis. Forbes et. al. (2004, 2006) create a system with two planar mirrors that can observe five silhouettes with different viewing angles. Extra silhouettes come from secondary reflections of mirrors. This system can calibrate itself with the geometry of mirrors, but it needs secondary reflections. Ho et. al. (2005,2009) work with a system that uses one mirror. Their method does not need prior knowledge, but they have to use two markers to calculate some information about 3D space. They use a volume-based approach. Ying at.al (2010) and Huang et. al. (2006) are used similar setups. Huang apply image based visual hull (Matusik et. al, 2000) in this setup.

5.3 Utilized Method

In this thesis, a system with three mirrors is used where viewpoints are evenly distributed. The aim of this work is to calculate a 3D reconstruction of an object in real time. To achieve this, parallel architectures are utilized. Since volume based SFS is much more favorable compered to surface based SFS in terms of parallelization, volume based approach is preferred.

The pseudocode of the algorithm used is given below. In this algorithm, a predetermined volume is divided into equal parts in Cartesian coordinates. Each sub-volume formed here is called a voxel. If each axis is divided into k equal parts, it means that k^3 voxels have been formed in the system. The resulting voxels can be written in the world coordinates ($\boldsymbol{P_w}$). In this algorithm, each voxel is projected onto silhouette images using calibration parameters, and it is checked whether it remains within the silhouette area. If the voxels are inside each silhouette area, then that voxel belongs to the object, but if it is outside of any silhouette, it does not belong to the object. Figure 5.3 shows a point cloud obtained using this algorithm.

Algorithm 1 Volume Based Shape from Silhouette

```
    Input: Silhouette images
 1  set division number    // that determines the number of voxels
 2  for each voxel in the axis X
 3      for each voxel in the axis Y
 4          for each voxel in the axis Z
 5              set Pw for each voxel // voxel's world coordinates
 6              for each silhouette picture
 7                  compute projection of Pw to the pictures
 8                  If projection of Pw is inside the silhouette
                        compute intersection of the projected points
                    end
 9              end
10              label the voxel
11          end
12      end
13  end
```

5.4 Texture Mapping

Texture mapping is a method that enhances visual richness of the computer-generated graphic or 3D model. There are numerous parameters of 3D data that can be enhanced by texture mapping (Heckbert, 1986).

- surface color mapping
- specular reflection
- normal vector perturbation ("bump mapping")
- specularity (the glossiness coefficient)
- transparency
- diffuse reflection
- shadows, surface displacement, and mixing coefficients
- local coordinate system ("frame mapping")

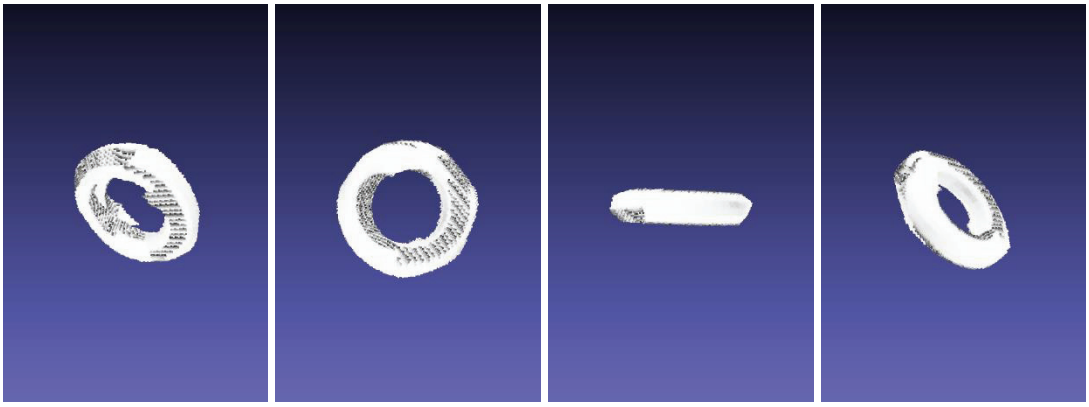

Figure 5. 3. Reconstruction of a torus

Surface color mapping is the main use for texture mapping. This operation assigns brightness values (or RGB values for color images) from a texture image which is usually acquired from photographic images of real objects. In our case, each voxels' texture information is acquired from the camera at the shortest distance. The pseudocode of this algorithm is given below. An example of texture mapping is shown in Figure 5.4.

Algorithm 2 Texture Mapping on 3D Object

```
    Input: RGB image
 1  compute  equidistant locations according to camera normal
 2  for each voxel inside of the object
 3      compute distance differences voxel and newly created points
 4      select the camera with shortest distance.
 5      take texture information from that camera
13  end
```

5.5 Results and Evaluations

In this thesis, a 3D reconstructing application was made with a camera and three mirrors. The system works with single snapshot taken with a real camera. This image, which has 1278x958 pixels, contains four silhouettes of the object. In this subsection, timing results for different voxel numbers of our implementation are given. In Table 5.1, 3D reconstruction results are shared without using texture mapping. In table 5.2, time results of 3D reconstruction with texture mapping are given. In addition, the results of several GPU-accelerated studies in the literature are shared (Table 5.3).

As seen in the tables above, the system can operate in real time (close to video frame rate speed) with up to 128^3 voxel numbers. Even with 256^3 voxel numbers, our implementation can run in under 1 second which is an acceptable speed for most inspection application.

In the Table 5.3, results of several GPU-accelerated studies are shared. Haro, (2012) and Matusik et.al (2002) use volume-based reconstruction among these algorithms.

According to these results, our implementation is comparable with state of art methods in terms of speed. Using only minimal amount of silhouettes captured from optimally separated viewing directions, allow us to achieve best overall quality from high speed 3D reconstruction.

Table 5. 1 Results of our implementation without texture mapping

Number of Voxel	Image Acquisition *(ms)*	Mem-cpy – Host to Device *(ms)*	Preprocessing *(ms)*	3D Reconstruction *(ms)*	Mem-cpy Device to Host *(ms)*
64^3	33	0.5	1.4	4	0.1
128^3	33	0.45	1.4	30	0.2
256^3	33	0.62	1.45	240	1.5
512^3	33	1.78	1.45	1780	11.4
1024^3	33	68.5	2.9	15987	97

Table 5. 2 Results with texture mapping

Number of Voxel	3D Reconstruction *(ms)*
64^3	5.9
128^3	40.4
256^3	297
512^3	2530
1024^3	20535

Table 5. 3 GPU-accelerated studies in the literature

	Silhouette Number	Image Resolution	Number of Voxel	Calculation Time (s)
Kolev (2012) - tori	20	640x480	-	3.54
Kolev (2012) - sow	27	1024x768	-	2.08
Haro (2012) - Girl	8	640x480	1038800	15
Matusik (2002)	4	-	-	0.047
Ladikos (2008)	16	1024x768	64^3	0.015
Ladikos (2008)	16	1024x768	128^3	0.026
Ladikos (2008)	16	1024x768	256^3	0.073

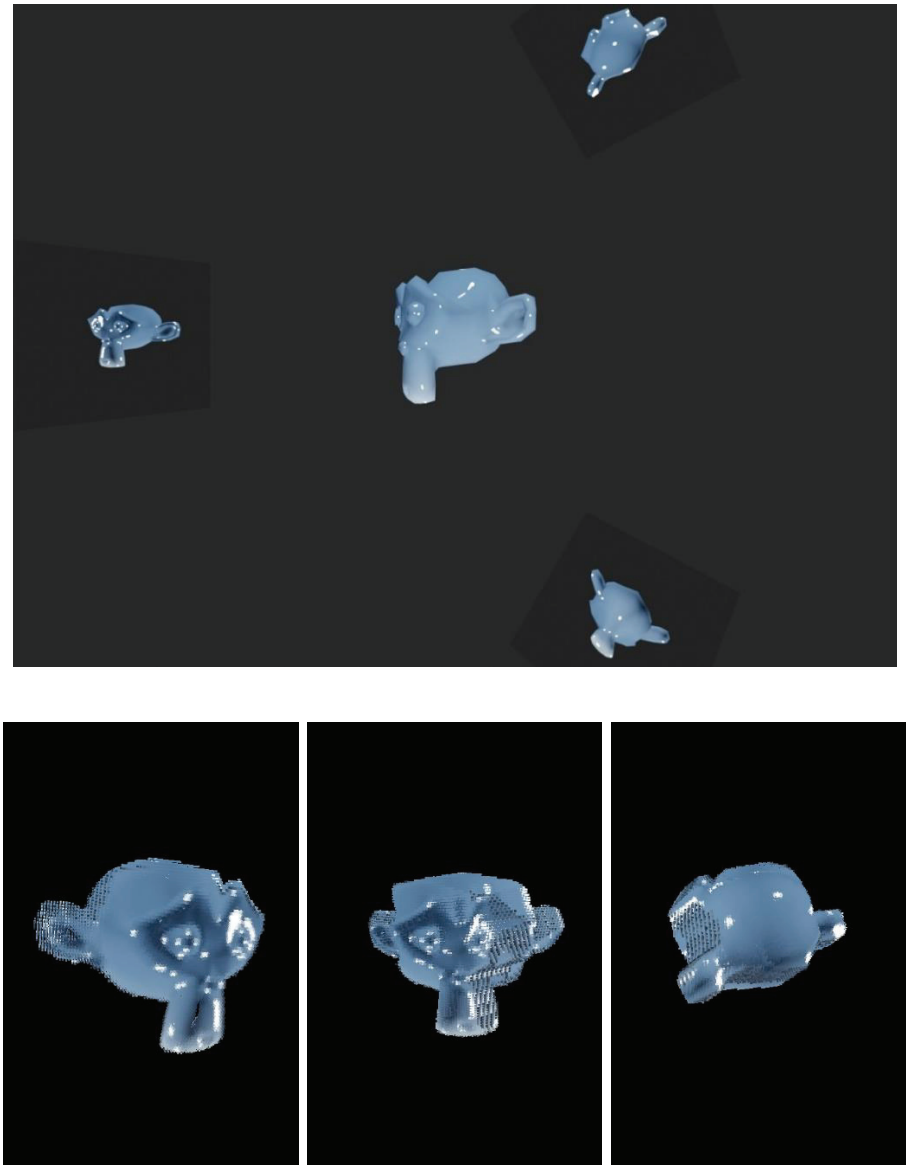

Figure 5. 4 Example image and its 3D reconstruction with texture

CHAPTER 6

CONCLUSIONS

The main goal of this thesis is to develop a real-time 3D reconstruction application that can complete full 3D reconstruction in a time period close to video frame rate. A simple and parallelizable SFS technique is employed for this purpose. One of the factors affecting the speed of SFS algorithms is the number of silhouettes used. A novel low cost system is introduced that captures silhouettes from optimal viewing directions in order to maintain this number at a minimum. A series of camera calibration algorithms that work well with out setup are described in detail.

The first part of this thesis involves system design. There are many SFS designs in the literature, but none of them have focus on viewing directions that are evenly distributed. A system design with an even distribution of viewing directions was created in our implementation, using regular tetrahedron structure. This design contributes to the system by allowing better and faster 3D reconstruction with fewer silhouettes.

The next part of the thesis is about the calibration of the real and virtual cameras of the system. At this stage, it is desired to design a calibration algorithm that can be easily reapplied when needed. First part of calibration is applied once to convert fisheye images to perspective images and to compute intrinsic parameters. Scaramuzza's omnidirectional camera calibration technique (2006-a) was employed in this step. Images are converted from the fisheye camera model to the perspective camera model with the coefficients of the inverse Taylor series obtained from this algorithm via backward mapping. Then, Zhang's calibration algorithm (2000) was used to obtain the intrinsic parameters. Finally, extrinsic parameters were obtained utilizing Guan's sphere camera calibration (2015) approach combined with Rudi's refinements (Penne et.al, 2019). This part of calibration can easily be reapplied when there is a change in positions due to environmental conditions.

The final stage of our implementation is the use of a SFS algorithm. In this step, many SFS algorithms in the literature were examined. In order to realize the main purpose of the thesis, a volume-based SFS algorithm is used, which is easily parallelizable.

The single camera, three mirror setup proposed in this thesis is a minimal setup to capture rough 3D geometry of most objects in sufficient detail. This low-cost setup is especially useful as a first step of 3D visual inspection applications. The proposed system can easily be extended by employing additional cameras and/or mirrors as well as other types of 3D reconstruction procedures. Flow diagram of our implementation can be seen in Figure 6.1.

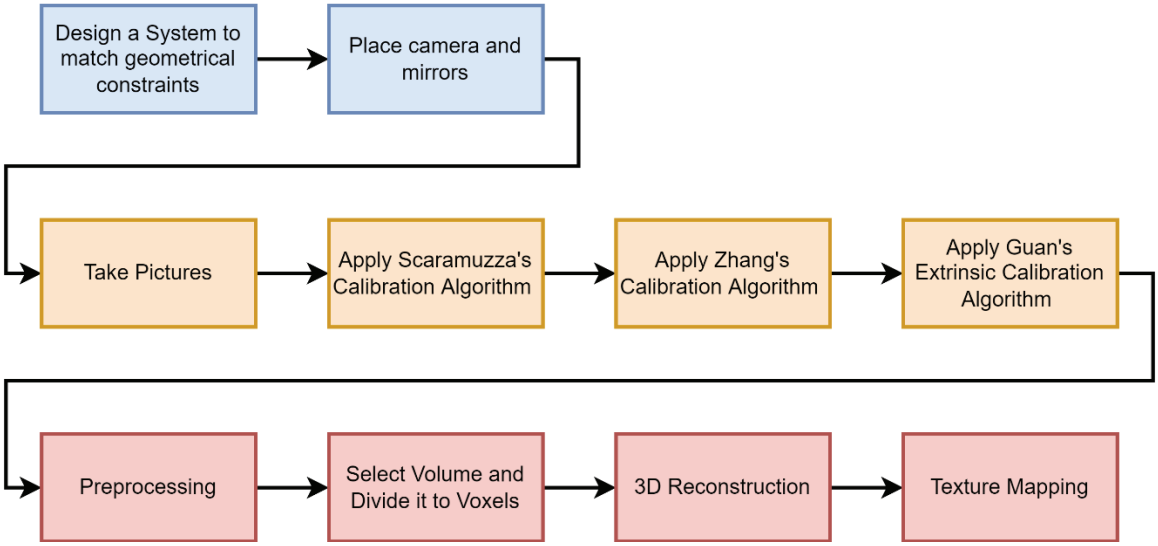

Figure 6. 1 Flow diagram of the study

REFERENCES

Agrawal, Motilal, and Larry S, Davis. "Camera calibration using spheres: A semi-definite programming approach.”. In Computer Vision, IEEE International Conference on (pp. 782–782).2003-a.

Agrawal, Motilal, and L, Davis. "Complete camera calibration using spheres: Dual space approach.”. In IEEE ICCV (pp. 782–789).2003-b.

Blender.
https://www.blender.org/features/ (accessed June 15,2021)

Butime, Julius, Inigo, Gutierrez, L Galo, Corzo, and C Flores, Espronceda. "3D reconstruction methods, a survey.”. In Proceedings of the First International Conference on Computer Vision Theory and Applications (pp. 457–463).2006.

Cheung, German KM, Simon, Baker, and Takeo, Kanade. "Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo.”. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. (pp. II–375).2003.

Cheung, Kong-man German, Simon, Baker, and Takeo, Kanade. "Shape-from-silhouette across time part ii: Applications to human modeling and markerless motion tracking".International Journal of Computer Vision 63, no.3 (2005): 225–245.

Cheung, Kong-man German, Simon, Baker, and Takeo, Kanade. "Shape-from-silhouette across time part i: Theory and algorithms".International Journal of Computer Vision 62, no.3 (2005): 221–247.

CUDA, CUDA Toolkit documentation (accessed May 10,2021)
https://docs.nvidia.com/cuda/

Dawson-Howe, Kenneth M, and David, Vernon. "Simple pinhole camera calibration".International Journal of Imaging Systems and Technology 5, no.1 (1994): 1–6.

Feng, Xiao-feng, and Di-fu, Pan. "A camera calibration method based on plane mirror and vanishing point constraint".Optik 154 (2018): 558–565.

Forbes, Keith. "Calibration, recognition, and shape from silhouettes of stones". (2007).

Forbes, Keith, Anthon, Voigt, and Ndimi, Bodika. "Visual hulls from single uncalibrated snapshots using two planar mirrors.". 2004.

Forbes, Keith, Fred, Nicolls, Gerhard, De Jager, and Anthon, Voigt. "Shape-from-silhouette with two mirrors and an uncalibrated camera.”. In European Conference on Computer Vision (pp. 165–178).2006.

Franco, Jean-Sébastien, and Edmond, Boyer. "Exact polyhedral visual hulls.”. In British Machine Vision Conference (BMVC'03) (pp. 329–338).2003.

Franco, Jean-Sébastien, and Edmond, Boyer. "Efficient polyhedral modeling from silhouettes".IEEE Transactions on Pattern Analysis and Machine Intelligence 31, no.3 (2008): 414–427.

Gluckman, Joshua, and Shree K, Nayar. "Catadioptric stereo using planar mirrors".International Journal of Computer Vision 44, no.1 (2001): 65–79.

Guan, Junzhi, Francis, Deboeverie, Maarten, Slembrouck, Dirk, Van Haerenborgh, Dimitri, Van Cauwelaert, Peter, Veelaert, and Wilfried, Philips. "Extrinsic calibration of camera networks using a sphere".Sensors 15, no.8 (2015): 18985–19005.

Haro, Gloria. "Shape from silhouette consensus".Pattern Recognition 45, no.9 (2012): 3231–3244.

Heckbert, Paul S. "Survey of texture mapping".IEEE computer graphics and applications 6, no.11 (1986): 56–67.

Hemayed, Elsayed E. "A survey of camera self-calibration.”. In Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003. (pp. 351–357).2003.

Hu, Bo, Christopher, Brown, and Randal, Nelson. "Multiple-view 3-D reconstruction using a mirror". (2005).

Hu, Bo. "It's all done with mirrors: Calibration-and-correspondence-free 3D reconstruction.”. In 2009 Canadian Conference on Computer and Robot Vision (pp. 148–154).2009.

Huang, Po-Hao, and Shang-Hon, Lai. "Contour-based structure from reflection.”. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (pp. 379–386).2006.

Kannala, Juho, and Sami S, Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses".IEEE transactions on pattern analysis and machine intelligence 28, no.8 (2006): 1335–1340.

Kanno, Isaku. "Fundamentals of piezoelectric thin films for microelectromechanical systems.”. In 2009 Nanostructures in Ferroelectric Films for Energy Applications. Elsevier. 237-255.

Kolev, Kalin, Thomas, Brox, and Daniel, Cremers. "Fast joint estimation of silhouettes and dense 3D geometry from multiple images".IEEE transactions on pattern analysis and machine intelligence 34, no.3 (2012): 493–505.

Kumar, Ram Krishan, Adrian, Ilie, Jan-Michael, Frahm, and Marc, Pollefeys. "Simple calibration of non-overlapping cameras with a mirror.”. In 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1–7).2008.

Kuzu, Yasemin, and Volker, Rodehorst. "Volumetric Modeling Using Shape from Silhouette". Fourth Turkish-German Joint Geodetic Dags. Vol. PAMI-5, no.2 (1983): 150–158.

Ladikos, Alexander, Selim, Benhimane, and Nassir, Navab. "Efficient visual hull computation for real-time 3D reconstruction using CUDA." . In 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (pp. 1–8).2008.

Lu, Yan, and Shahram, Payandeh. "On the sensitivity analysis of camera calibration from images of spheres".Computer Vision and Image Understanding 114, no.1 (2010): 8–20.

Matusik, Wojciech, Chris, Buehler, Ramesh, Raskar, Steven J, Gortler, and Leonard, McMillan. "Image-based visual hulls.”. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (pp. 369–374).2000.

Matusik, Wojciech, Chris, Buehler, Leonard, McMillan, and Steven J, Gortler. "An efficient visual hull computation algorithm".Tech. Rep., MIT LCS Technical Memo 623, MIT Laboratory for Computer Science (2002).

Mei, Christopher, and Patrick, Rives. "Single viewpoint omnidirectional camera calibration from planar grids.". In Proceedings 2007 IEEE International Conference on Robotics and Automation (pp. 3945–3950).2007.

Michoud, Brice, Erwan, Guillou, and Saida, Bouakaz. "Shape From Silhouette: Towards a Solution for Partial Visibility Problem.". In Eurographics (Short Presentations) (pp. 13–16).2006.

Olsson, Karin, and Therese, Persson. "Shape from silhouette scanner." (2002).

Penne, Rudi, Bart, Ribbens, Luc, Mertens, and Paul, Levrie. "What does one image of one ball tell us about the focal length?". In International Conference on Advanced Concepts for Intelligent Vision Systems (pp. 501–509).2015.

Penne, Rudi, Bart, Ribbens, and Pedro, Roios. "An exact robust method to localize a known sphere by means of one image".International Journal of Computer Vision 127, no.8 (2019): 1012–1024.

Rodrigues, Rui, Joao P, Barreto, and Urbano, Nunes. "Camera pose estimation using images of planar mirror reflections.". In European Conference on Computer Vision (pp. 382–395).2010.

Scaramuzza, Davide, Agostino, Martinelli, and Roland, Siegwart. "A toolbox for easily calibrating omnidirectional cameras.". In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 5695–5701).2006-a.

Scaramuzza, Davide, Agostino, Martinelli, and Roland, Siegwart. "A flexible technique for accurate omnidirectional camera calibration and structure from motion.". In Fourth IEEE International Conference on Computer Vision Systems (ICVS'06) (pp. 45–45).2006-b.

Siudak, Mariusz, and Przemyslaw, Rokita. "A survey of passive 3d reconstruction methods on the basis of more than one image.". Machine Graphics & Vision 23 (2014).

Szeliski, Richard. "Rapid octree construction from image sequences".CVGIP: Image understanding 58, no.1 (1993): 23–32.

Szeliski, Richard. Computer vision: algorithms and applications.Springer Science & Business Media, 2010.

Takahashi, Kosuke, Shohei, Nobuhara, and Takashi, Matsuyama. "A new mirror-based extrinsic camera calibration using an orthogonality constraint.”. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1051–1058).2012.

Tarini, Marco, Marco, Callieri, Claudio, Montani, Claudio, Rocchini, Karin, Olsson, and Therese, Persson. "Marching Intersections: An Efficient Approach to Shape-from-Silhouette.”. In VMV (pp. 283–290).2002.

Urban, Steffen, Jens, Leitloff, and Stefan, Hinz. "Improved wide-angle, fisheye and omnidirectional camera calibration".ISPRS Journal of Photogrammetry and Remote Sensing 108 (2015): 72–79.

VTK. Visualization Toolkit documentation https://vtk.org/doc/nightly/html/ (accessed October 10,2021)

Wong, Kwan-Yee Kenneth, Guoqiang, Zhang, and Zhihu, Chen. "A stratified approach for camera calibration using spheres".IEEE Transactions on Image Processing 20, no.2 (2010): 305–316.

Yin, Wei, Shijie, Feng, Tianyang, Tao, Lei, Huang, Song, Zhang, Qian, Chen, and Chao, Zuo. "Calibration method for panoramic 3d shape measurement with plane mirrors".Optics express 27, no.25 (2019): 36538–36550.

Ying, Xianghua, Kun, Peng, Ren, Ren, and Hongbin, Zha. "Geometric properties of multiple reflections in catadioptric camera with two planar mirrors." . In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 1126–1132).2010.

Yous, Sofiane, Hamid, Laga, Masatsugu, Kidode, and Kunihiro, Chihara. "Gpu-based shape from silhouettes.”. In Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia (pp. 71–77).2007.

Zhang, Hui, Guoqiang, Zhang, and K-YK, Wong. "Camera calibration with spheres: linear approaches.”. In IEEE International Conference on Image Processing 2005 (pp. II–1150).2005.

Zhang, Hui, K Wong, Kwan-yee, and Guoqiang, Zhang. "Camera calibration from images of spheres".IEEE Transactions on Pattern Analysis and Machine Intelligence 29, no.3 (2007): 499–502.

Zhang, Zhengyou. "A flexible new technique for camera calibration".IEEE Transactions on pattern analysis and machine intelligence 22, no.11 (2000): 1330–1334.

APPENDIX A

SYSTEM DESIGN FRAGMENTS

Geometry of the Tetrahedron

Tetrahedrons are the simplest polyhedrons. They are also known as triangular pyramids. They have four corners and six edges, and they consist of four triangular surfaces. Tetrahedrons have two special spheres that are called the insphere and circumsphere (Figure A.1). Inspheres are tangent to the surfaces of the tetrahedron, and circumspheres contain all four corners.

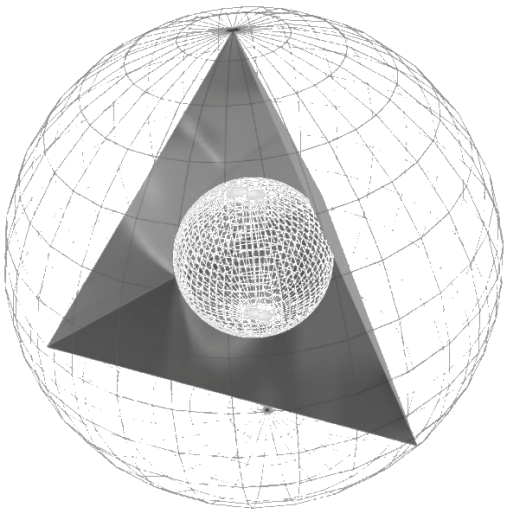

Figure A. 1 Regular tetrahedron and its circumsphere and insphere

There are some special tetrahedrons. One of them is regular tetrahedron which is all faces are equilateral triangle. The geometry of regular tetrahedrons is used while designing the system.

A : length of the tetrahedron edge.

H : height of the tetrahedron volume (i.e. distance from a corner to the midpoint of the opposite surface)

d_{oc}: length between one of the corner and center of the volume.

α : angle between A and H

$$H = A\sqrt{\frac{2}{3}} \tag{A.1}$$

$$d_{oc} = H\,\frac{3}{4} \tag{A.2}$$

$$\alpha = \operatorname{acos}\left(\frac{H}{A}\right) = 35.2644° \tag{A.3}$$

Falling axis

In this system there is a need for falling axis because the objects to be reconstructed are needed to enter common field of area. They must fall along the Z axis without crashing any mirror or camera. This can be achieved by joining midpoints of two opposite sides of the tetrahedron. The tetrahedron is rotated to match this line with the (vertical) Z axis, which can be seen in Figure A.2.

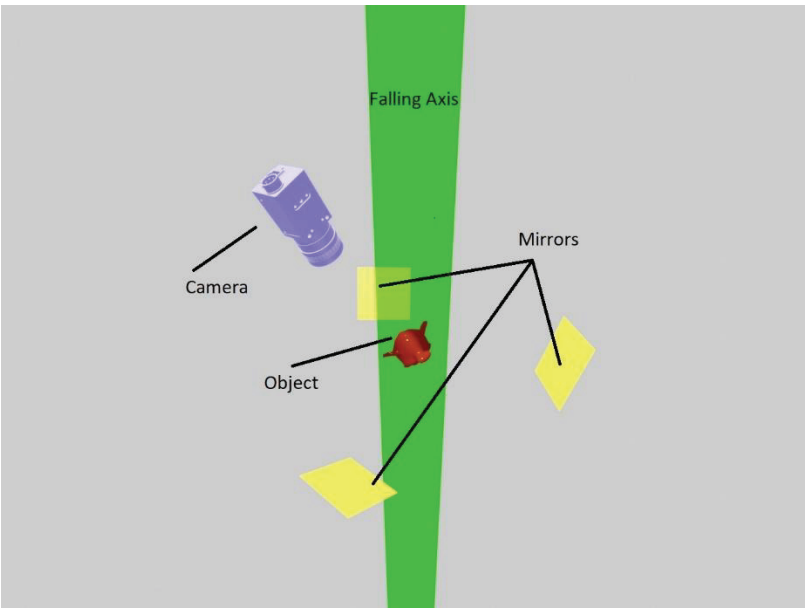

Figure A. 2 Falling axis and rotated system

Trigger Mechanism

A trigger mechanism is needed to detect a falling object at the correct time by the camera. The trigger mechanism consists of a sensor, electronic circuit, and microcontroller. The block diagram of this mechanism is given in Figure A.3.

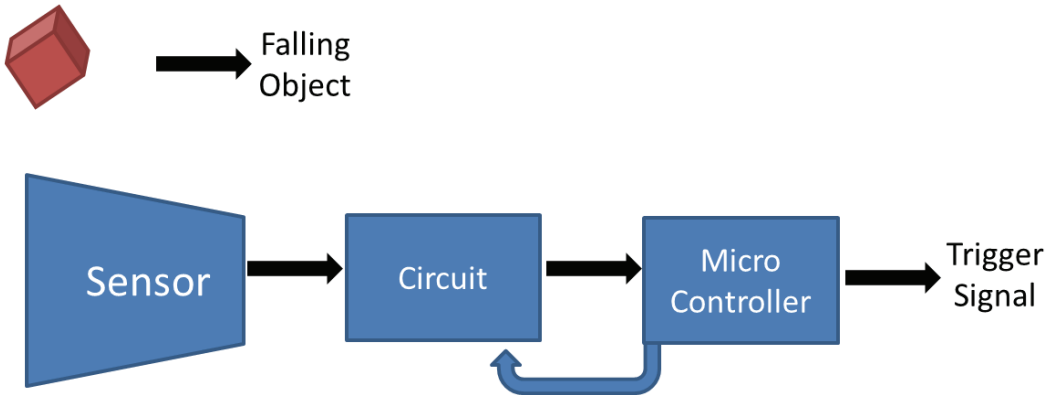

Figure A. 3 Block diagram of the trigger mechanism

Sensor and the microcontroller are introduced in the chapter 2. In this subsection, circuit and the necessary code will be introduced.

The piezoelectric film sensor cannot produce a signal that the microcontroller can understand. Therefore, there is a need to design a circuit. This circuit consists of a full-wave rectifier, peak detector, and comparator as shown in Figure A.4.

The output of the piezoelectric film sensor is a time-varying signal. This signal first enters the full-wave rectifier. The full-wave rectifier aims to transfer the signals from the negative side to the positive side. The signal coming out of the full-wave rectifier enters the peak detector and finds its maximum value. This signal passes through a comparator with a 2.5V threshold and becomes understandable to the microprocessor.

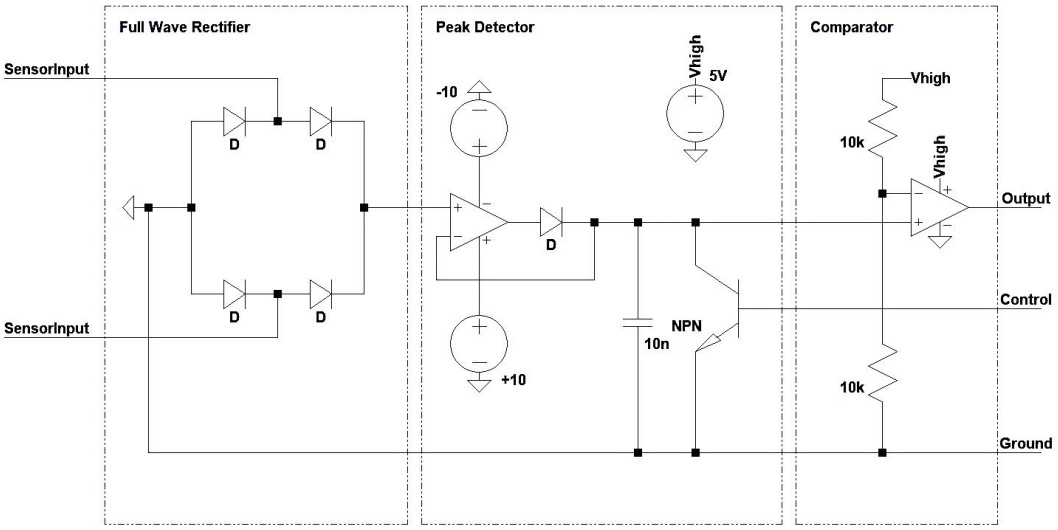

Figure A. 4 The circuit between sensor and microcontroller

The control output of the circuit also goes to the microcontroller. The purpose of the control output is to make the circuit usable again. When the necessary signal is given to the control pin, the capacitor of the peak detector is discharged and can be and reused.

Design elements

In this section some of design elements, that created in Blender, are shown. Figure A.5 shows a case for the mirror. Figure A.6 demonstrate some pieces for illumination parts for real camera. LED oriantation pieces and mirror holders can be seen in Figure A.7. Finnaly, assembled mirror holder is shown in Figure A.8.

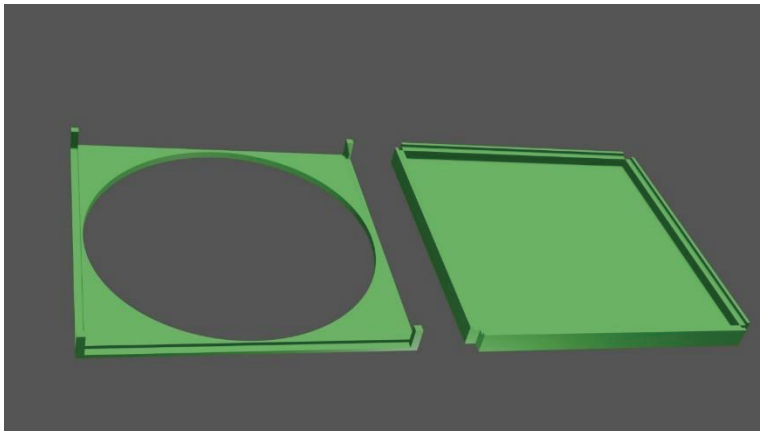

Figure A. 5 Mirror-holder case bottom (right) and top (left)

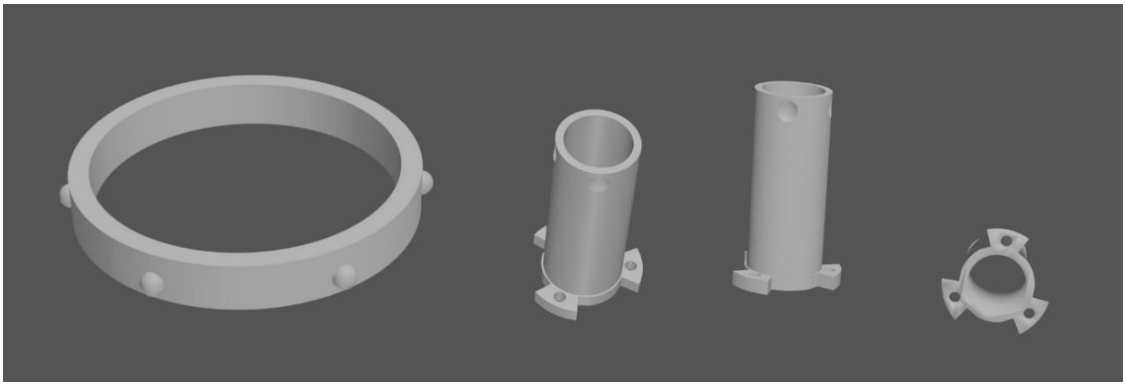

Figure A. 6 Camera lighting ring (left) and LED-holder (right)

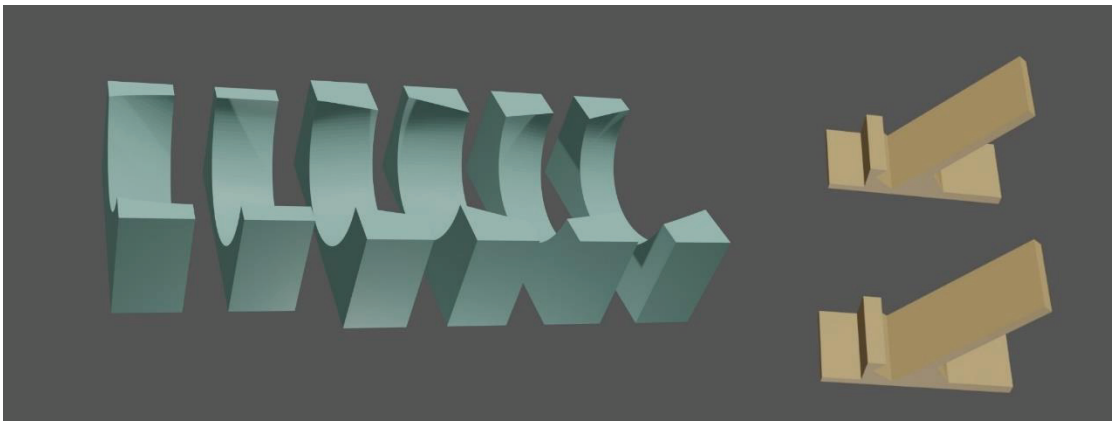

Figure A. 7 LED-holder orientation pieces (left) and mirror-holder legs (right)

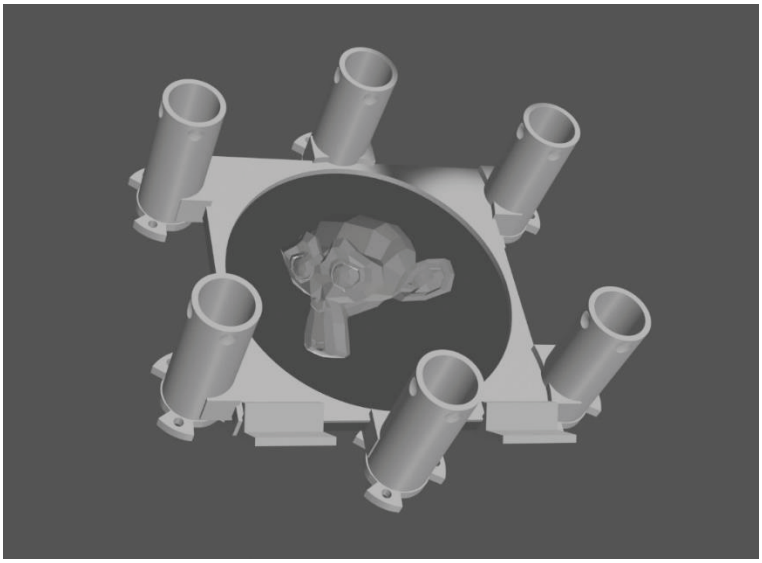

Figure A. 8 Assembled mirror-holder