Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: tos@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-68 February 15, 2002

Adaptive 3D Modeling of Objects by Combining Shape from Silhouette and Shape from Structured Light

Diploma thesis

Srdan Tosovic

Abstract

This thesis proposes a method of three-dimensional reconstruction of objects using a combination of two different methods, Shape from Silhouette and Shape from Structured Light, focusing on reconstruction of archaeological vessels. Shape from Silhouette is a method suitable for reconstruction of objects with handles, whereas it is unable to reconstruct concavities on an object's surface, such as inside of a bowl. Shape from Structured Light can reconstruct such concavities, but it often creates incomplete models because of camera and light occlusions. The purpose of combining these two methods is to overcome the weaknesses of one method through the strengths of the other, making it possible to construct complete models of arbitrarily shaped objects. The construction is based on multiple views of an object using a turntable in front of stationary cameras. The method is adaptive, because it automatically selects a subset of possible views, guided by the complexity of the object modeled. Results of the algorithm developed are presented for both synthetic and real objects.

Kurzfassung

Diese Diplomarbeit stellt ein Verfahren zur dreidimensionalen Erfassung von Objekten vor, das auf zwei im Bereich der Computer Vision bekannten Methoden basiert: Shape from Silhouette und Shape from Structured Light. Bei Shape from Silhouette wird das 3D-Modell aus mehreren Bildern gewonnen, welche die Silhouetten des Objektes aus verschiedenen Blickwinkeln darstellen. Dieses Verfahren eignet sich für Modellierung von konvexen Objekten, sowie Objekten mit Henkeln. Konkavitäten, wie z.B. das Innere einer Schüssel, können mit dieser Methode nicht rekonstruiert werden. Deswegen wird Shape from Silhouette mit Shape from Structured Light kombiniert. Bei dieser Technik werden unterschiedliche, im Vorhinein definierte Lichtmuster auf das Objekt projiziert. Aus den aufgenommenen Bildern und der bekannten Geometrie zwischen dem Lichtprojektor und der Kamera werden die dreidimensionalen Koordinaten der Punkte auf der Objektoberfläche berechnet. Mit diesem Verfahren können theoretisch beliebige, also auch konkave Flächen erfaßt werden. In der Praxis sind aber mit dieser Methode erzeugte Flächenmodelle oft unvollständig, wegen Licht- und Kameraverdeckungen. Das vorgestellte, kombinierte Verfahren versucht die Probleme der beiden zugrundeliegenden Methoden aufzuheben und Objekte beliebiger Form vollständig zu erfassen. Experimente mit synthetischen und realen Objekten werden beschrieben und die Ergebnisse präsentiert. Verschiedene Ansichten der Objekte werden durch die Drehung eines Rotationstellers gewonnen. Es wurde eine Methode der automatischen Auswahl der Ansichten entwickelt, weswegen das vorgestellte Modellierungsverfahren als adaptiv bezeichnet wird.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The 3D modeling method proposed in this thesis has been done as part of the project *Computer Aided Classification of Ceramics* [SM96, KS99a, AKK+01], which is performed by the Pattern Recognition and Image Processing Group at the Institute of Computer Aided Automation at the Vienna University of Technology in cooperation with the Institute of Classical Archeology at the University of Vienna, with the goal of providing an objective and automated method for classification and reconstruction of archaeological pottery.

Three-dimensional reconstruction of archaeological vessels and sherds is interesting for archaeologists for several reasons. The models can be analyzed by archaeologists without having physical access to the excavation site, or be exhibited in virtual museums. The volume of a vessel can be estimated by calculating the volume of its 3D model, which allows more precise classification [OTV93]. Furthermore, by intersecting a 3D model of a sherd with an appropriate plane a nearly perfect sherd profile can be extracted. Profiles are commonly used by archaeologists for classification of sherds [OTV93]. The surface of a 3D model of a vessel can also be unwrapped, which can help archaeologists illustrate its decoration.

At the PRIP Group, there have been performed many works related to classification and reconstruction of archaeological pottery. Here we will mention the most recent ones: surface reconstruction using Shape from Structured Light based on projection of laser planes onto the object [Lis99, LS00]; surface reconstruction based on registration of range images of the front and the back view of the object [Kam99, KS99b]; volume reconstruction based on Shape from Silhouette [Tos00, TS01]; classification of sherds based on profile primitives [KSC01, SKS01].

The method proposed in this work is a combination of Shape from Silhouette and Shape from Structured Light, using a turntable to obtain multiple views of an object. The works of Tosovic [Tos00] and Liska [Lis99] were used as a base for the combined method. Both of the underlying methods have their strengths and weaknesses. The idea behind combining them is to overcome the weaknesses of one method by using the strengths of the other and build models more precise and accurate than the models built

using one of the methods only.

In addition to 3D modeling of objects, we also developed a method for automatic selection of views, i.e., Next View Planning [TAT95]. However, this has been a secondary goal of the thesis, made after the work on modeling was done. Therefore, Next View Planning is introduced and described in a separate section (Section 6).

The remainder of this chapter gives an overview of Shape from Silhouette, Shape from Structured Light and the related works, followed by an outline of the thesis structure.

1.1 Shape from Silhouette

Shape from Silhouette is a method of automatic construction of a 3D model of an object based on a sequence of images of the object taken from multiple views, in which the object's silhouette represents the only interesting feature of an image [Sze93, Pot87]. The object's silhouette in each view (Figure 1.1a) corresponds to a conic volume in 3D space (Figure 1.1b). A 3D model of an object (Figure 1.1c) can be obtained by intersecting the conic volumes, which is also called *Space Carving* [KS00]. Multiple views of the object can be obtained either by moving the camera around the object or by moving the object inside the camera's field of view. In our approach the object rotates on a turntable in front of a stationary camera.

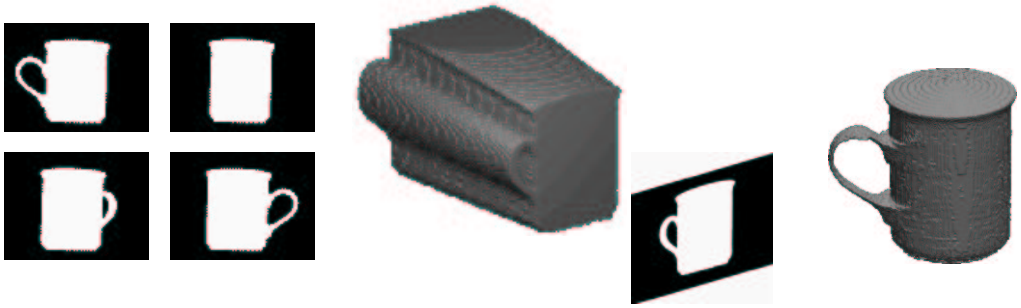


Figure 1.1: Image silhouettes (a), a conic volume (b) and the final model (c)

Shape from Silhouette can be applied on objects with variety of shapes, including objects with handles, like the cup in Figure 1.1c and like many archaeological vessels and sherds. However, concavities on an object's surface remain invisible for this method, making it unusable for reconstruction of the inside of a bowl or a cup (Figure 1.1c) or the inner side of a sherd. Therefore, another method, Shape from Structured Light, is used to discover the concavities.

There have been many works on Shape from Silhouette based construction of 3D models of objects, in literature also called Visuall Hull construction. Baker [Bak77] used

silhouettes of an object rotating on a turntable to construct a wire-frame model of the object. Martin and Aggarwal [MA83] constructed volume segment models from orthographic projection of silhouettes. Chien and Aggarwal [CA86] constructed an object's octree model from its three orthographic projections. Veenstra and Ahuja [VA86] extended this approach to thirteen standard orthographic views. Potmesil [Pot87] created octree models using arbitrary views and perspective projection. For each of the views he constructs an octree representing the corresponding conic volume (Figure 1.1b) and then intersects all octrees. In contrast to this, Szeliski [Sze93] first creates a low resolution octree model quickly and then refines this model iteratively, by intersecting each new silhouette with the already existing model. The last two approaches project an octree node into the image plane to perform the intersection between the octree node and the object's silhouette. Srivastava and Ahuja [SA90] in contrast, perform the intersections in 3D-space. Niem [Nie94] uses pillar-like volume elements (pillars) rather than octree for model representation and compares the complexity of his approach with Potmesil's [Pot87] and Szeliski's [Sze93] approach, suggesting that his algorithm requires far less intersection tests. De Bonet and Viola [BV99] extended the idea of voxel reconstruction to transparent objects by introducing the Roxel algorithm — a responsibility weighted 3D volume reconstruction. Wong and Cipolla [WC01] use uncalibrated silhouette images and recover the camera positions and orientations from circular motions. In the recent years there have been also Shape from Silhouette approaches based on video sequences [DF99, BL00]. The work of Szeliski [Sze93] was used as a base for the Shape from Silhouette part of the method presented in this thesis, initially developed in [Tos00].

1.2 Shape from Structured Light

Shape from Structured Light is a method which constructs a surface model of an object based on projecting a sequence of well defined light patterns onto the object. The patterns can be in the form of coded light stripes [Kam99] or a ray or plane of laser light [Lis99]. For every pattern an image of the scene is taken. This image, together with the knowledge about the pattern and its relative position to the camera are used to calculate the coordinates of points belonging to the surface of the object. This process is also called *active triangulation* [Bes88, DT96], illustrated in Figure 1.2. If the geometry between the laser plane and the image is known, then each 2D image point belonging to the laser line corresponds to exactly one 3D point on the surface of the object.

Shape from Structured Light method used in our approach is based on projection of laser planes onto the object (Figure 1.3a). 3D points obtained through active triangulation using all views represent a cloud of points belonging to the object's surface (Figure 1.3b). This cloud of points can be used to create a smooth surface representation (Figure 1.3c).

A strength of Shape from Structured Light is that it can reconstruct any kind of concavities on the surface of the object (see the top of the amphora in Figures 1.3b and 1.3c), as long as the projected light reaches these concavities and the camera detects it. However,

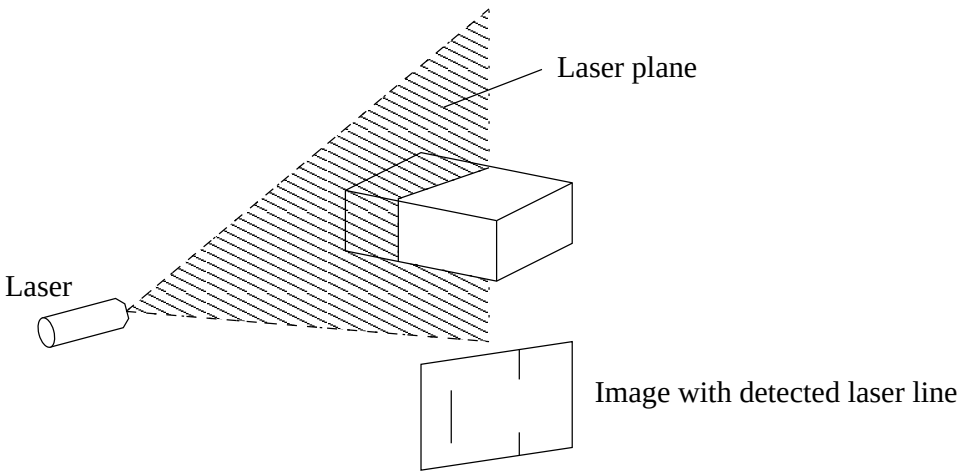


Figure 1.2: Active triangulation

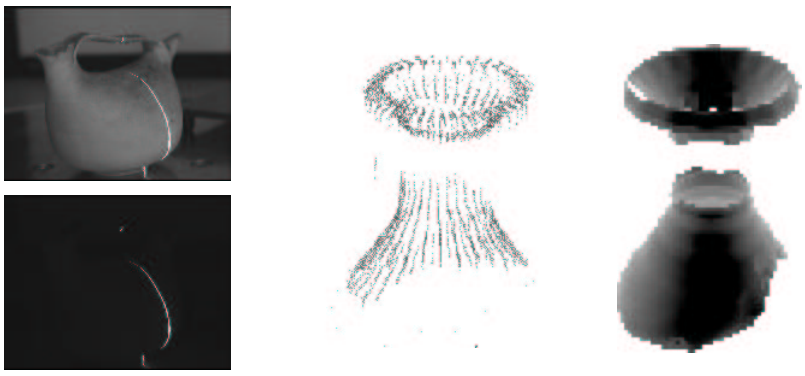


Figure 1.3: Projection of laser plane (a), cloud of points (b) and reconstructed surface (c)

this method often suffers from camera and light occlusions [Lis99], resulting in incomplete surface models (like the neck of the amphora in Figure 1.3c).

Most laser light based Shape from Structured Light methods use a camera, a calibrated laser ray or plane and a motion platform — usually a linear slide or a turntable. Borgese et al. [BFB+98] use a pair of standard video cameras, a laser pointer, and a special hardware that lets the laser spot be detected with high reliability and accuracy. Takatsuka et al. [TWVC99] built a range scanner consisting of one camera and a laser pointer, to which three LEDs are attached. The camera captures the image of spots (one from the laser, and the others from LEDs), and triangulation is carried out using the camera's viewing direction and the optical axis of the laser. By using a laser pointer, both of these methods [BFB+98, TWVC99] obtain a single surface point at each step, which implies a slow, sparse sampling of the surface. Liska [Lis99, LS00] uses two lasers aligned to project the same plane, a camera and a turntable. Using two lasers eliminates some

of the light occlusions but not the camera occlusions, resulting in incomplete models for many objects (images of the model of the amphora in Figures 1.3b and 1.3c were taken from [Lis99]). Park et al. [PDK01] built a DSLS (Dual Beam Structured Light) scanner, which uses two lasers and a camera mounted on a linear slide. The planes projected by the two lasers never overlap in the camera view, resulting in denser range images and a reduced number of view registrations required for 3D modeling. Davis and Chen [DC01] use two calibrated fixed cameras viewing a static scene and an uncalibrated laser plane which is freely swept over the object. Focus of their research is also to minimize the cost of the equipment and therefore they actually use a single camera and a system of mirrors arranged to split the camera's view into two virtual views. Other Shape from Structured Light approaches similar to projection of a laser plane use an ordinary light source and some kind of wand which is placed in front of the light source, creating a shadow on the object, thus defining a plane in 3D space. Bouguet and Perona [BP98] use a checkerboard as the ground plane and move a pencil in front of a desk-lamp, which illuminates the object. The object's 3D surface coordinates are extracted by tracking the pencil shadows on the object and the ground plane. Fisher et al. [FARW99] use a similar approach, but, beside tracking the wand shadow on the object, they track the wand itself instead its ground plane shadow. Recently there are also projects on reconstruction on large statues with high precision and high cost equipment, such as modeling of historic heritage in Japan [NSI99] and the Digital Michelangelo Project [LPC+00], where the models achieve the resolution of as fine as 0.05 mm. The work of Liska [Lis99] was used as a guide for the Shape from Structured Light part of the approach presented in this thesis.

There are not many works on combination of Shape from Silhouette and Shape from Structured Light known to the author of this thesis. Immersion Corporation [Imma] makes a commercial product called LightScribe [Immb] which combines Shape from Silhouette, Shape from Structured Light and texture mapping for generation of fully textured models of real objects, priced at about ten thousand dollars.

1.3 Thesis Structure

This document is structured as follows: Chapter 2 presents the mathematical background necessary to understand all geometrical transformations performed by the modeling algorithm, such as camera models and geometrical transformations between 2D and 3D coordinate systems. Chapter 3 describes our acquisition system, its configuration and calibration. Chapter 4 discusses 3D modeling in general and possible approaches for combining Shape from Silhouette with Shape from Structured Light, followed by a detailed description of our combined approach in Chapter 5. Chapter 6 introduces Next View Planning, first in general and then describing our approach for selection of the next view. The experimental results with synthetic and real objects, as well as with Next View Planning, are presented in Chapter 7. Finally, Chapter 8 concludes the thesis by giving a summary and an outlook to the possible future work.

Chapter 2

Mathematical Background

This chapter introduces the mathematical background for the 3D modeling approach presented in this thesis. Beginning from an overview of the pinhole camera model in Section 2.1, Tsai's parameterization of this model is described in Section 2.2, followed by a description of the process of calculating image coordinates of any 3D point in space in Section 2.3. Section 2.4 gives a brief overview of the camera calibration techniques. Finally, Section 2.5 discusses a 2D plane in 3D space and its parameterization, followed by a summary in Section 2.6.

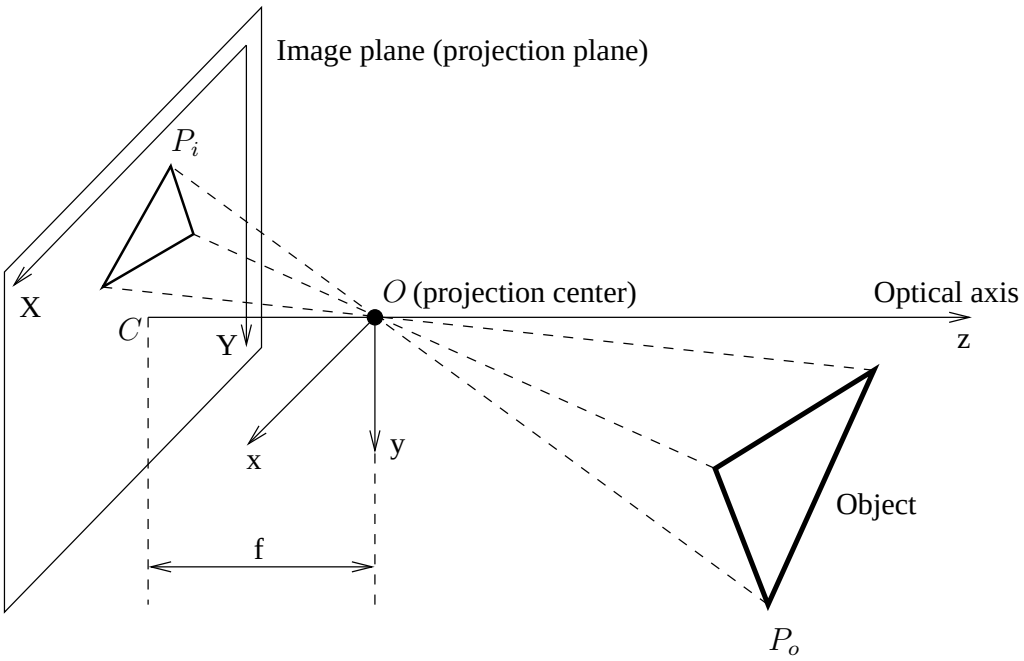
2.1 Pinhole Camera Model

The pinhole camera model [Nal93] assumes an ideal camera, i.e., a camera without any lens distortions. The camera is assumed to have an infinitely small hole through which the light enters before forming an inverted image on the camera's image plane. The pinhole camera model is also called perspective camera model because the creation of an image in the image plane can be described through perspective projection completely [Fau93]. Perspective projection is the projection of a three-dimensional object onto a two-dimensional surface by straight lines that pass through a single point (Figure 2.1). It is completely defined by choosing a perspective projection center and a projection plane. For a pinhole camera, the projection center is the hole where the light enters and the projection plane is the camera's image plane.

If we assume that the coordinate system is rooted in the perspective center and that its x-y plane is parallel to the image plane with the distance f (as shown in Figure 2.1), then the image coordinates (x_i, y_i) are related to the object coordinates (x_o, y_o, z_o) by:

$$x_i = \frac{f}{z_o} x_o \text{ and } y_i = \frac{f}{z_o} y_o \tag{2.1}$$

In case z_o is zero, both x_i and y_i are equal zero. Using *homogeneous* form [Nal93], a 2D or 3D transformation consisting of a combination of rotation, scaling and translation can



$P_o = (x_o, y_o, z_o)$
$P_i = (x_i, y_i)$ or (X_i, Y_i)
$C = (C_x, C_y)$

Figure 2.1: Pinhole camera model and perspective projection

be described by a single matrix. In homogeneous form, Equation 2.1 can be written as:

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{z_o} & 0 & 0 & 0 \\ 0 & \frac{f}{z_o} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_o \\ y_o \\ z_o \\ 1 \end{pmatrix} \tag{2.2}$$

The coordinates (x_i, y_i) are given in the coordinate system rooted in the perspective center. This system is called *camera coordinate system*. If we want to calculate the corresponding coordinates (X_i, Y_i) in the *image coordinate system* (X-Y in Figure 2.1), we have to take into account four additional parameters:

- C_x, C_y — the image coordinates of the point C in Figure 2.1, which is called the image principal point — it is the intersection of the camera's optical axis with the image plane.
- d_x, d_y — the distance between two neighboring sensor elements of the camera in x and y direction.

Given these parameters and the camera coordinates (x_i, y_i) of the point P_i from Figure 2.1, its image coordinates (X_i, Y_i) can be calculated as follows:

$$X_i = \frac{x_i}{d_x} + C_x \text{ and } Y_i = \frac{y_i}{d_y} + C_y \tag{2.3}$$

or in the homogeneous form:

$$\begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{d_x} & 0 & C_x \\ 0 & \frac{1}{d_y} & C_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{2.4}$$

By replacing the vector $(x_i, y_i, 1)^T$ in Equation 2.4 with the right hand side of the Equation 2.2 we can build the matrix transforming the object coordinates (in camera coordinate system) (x_o, y_o) into image coordinates (X_i, Y_i) of a pinhole camera:

$$\begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{z_o d_x} & 0 & 0 & C_x \\ 0 & \frac{f}{z_o d_y} & 0 & C_y \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_o \\ y_o \\ z_o \\ 1 \end{pmatrix} \tag{2.5}$$

2.2 Tsai's Camera Model

In Section 2.1 we discussed the transformation of the camera coordinates into the image coordinates. In a real situation we define a certain world coordinate system independent from where the camera is placed and we want to establish the relation between the world coordinates and the computer image coordinates (Figure 2.2). The establishing of such a relation is called *camera calibration* (see also Section 2.4). The relation between the world and the camera coordinate system defines the position of the camera in the world 3D space and it is called camera *external* orientation. The relation between the camera and the computer image coordinate system depends on the internal parameters of the camera and the frame grabber used and it is called camera *internal* orientation. The transformations in Section 2.1 also ignore any possible lens distortions. This can be sufficient for some applications, but if higher accuracy is required, lens distortion has to be taken into account.

Tsai's camera model [Tsa86] is a pinhole based model which defines the transformation between the desired world coordinate system and the computer image coordinate system and takes into account nonlinear effects in the process of image acquisition such as radial lens distortion. It is the model used for the 3D modeling approach presented in this thesis. Tsai's model has 16 parameters — 6 external, 6 internal and 4 so-called *fixed* internal parameters.

The six *external* (also called extrinsic) parameters define the external orientation of the camera, i.e., the transformation between the world coordinate system and the camera coordinate system (see Figure 2.2). This transformation can uniquely be defined as a 3D rotation around the origin followed by a 3D translation. Each of these transformations is defined by three parameters:

- ψ, ϕ, θ — Euler angles defining the 3D rotation: roll ψ, pitch ϕ and yaw θ
- T_x, T_y, T_z — the components of the vector T from Figure 2.2 defining the 3D translation.

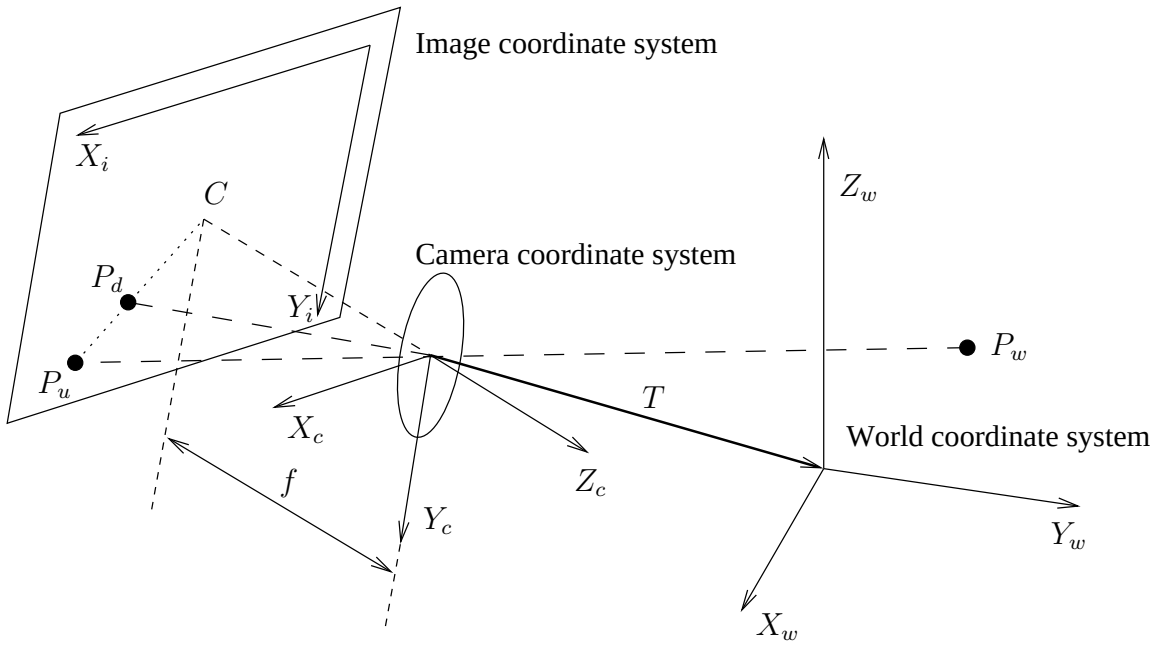


Figure 2.2: Coordinate systems in Tsai's camera model

The ten *internal* (also called intrinsic) parameters define the transformation between the camera coordinate system and the image coordinate system. This transformation can be viewed as the transformation of the camera coordinates into the undistorted image coordinates described in Section 2.1 and characterized by Equation 2.5, followed by an additional nonlinear transformation as a result of the radial lens distortion. The point P_u in Figure 2.2 is the undistorted perspective projection of the point P_w calculated according to Equation 2.5. The radial lens distortion moves this point to the point P_d along the line $\overline{CP_u}$ in Figure 2.2. How P_d is derived from P_u is described in detail in Section 2.3.

Tsai's camera model defines the following internal parameters:

- f — effective focal length of the camera, i.e., the distance between the optical center and the image plane (see Figure 2.2).
- κ_1 — radial lens distortion coefficient.
- κ_2 — tangential lens distortion coefficient. According to [Tsa86] its value is negligible compared to the radial distortion and therefore it is usually ignored.
- C_x, C_y — computer image coordinates of the intersecting point of the optical axis of the camera with the image plane, i.e., the principal point (C in Figure 2.2).
- s_x — scale factor to account for any uncertainty in the frame grabber's resampling of the horizontal scanline.
- N_{cx} — number of sensor elements in camera's x direction.

- N_{fx} — number of pixels in frame grabber's x direction.
- d_x — x dimension of camera's sensor element (in mm).
- d_y — y dimension of camera's sensor element (in mm).

The last four parameters are called *fixed* because they never change — they should be provided by the camera and frame grabber manufacturers.

2.3 World-to-image Transformation

As described in [Tsa86], the transformation from the world coordinate system to the computer image coordinate system is performed in 4 steps (as illustrated in Equation 2.6):

1. Rigid body transformation from the world coordinate system to the camera coordinate system, i.e., calculation of the camera coordinates $(x_{w_{cam}}, y_{w_{cam}}, z_{w_{cam}})$ of the point P_w in Figure 2.2 from its world coordinates $(x_{w_{wrl}}, y_{w_{wrl}}, z_{w_{wrl}})$.
2. Perspective projection of the camera coordinates of P_w into the image plane, i.e., obtaining the undistorted camera coordinates $(X_{u_{cam}}, Y_{u_{cam}})$ of the point P_u from Figure 2.2.
3. Radial lens distortion of P_u, i.e., calculating camera coordinates $(X_{d_{cam}}, Y_{d_{cam}})$ of the point P_d.
4. Calculation of the image coordinates $(X_{d_{img}}, Y_{d_{img}})$ of the point P_d in the image coordinate system.

$$\begin{pmatrix} x_{w_{wrl}} \\ y_{w_{wrl}} \\ z_{w_{wrl}} \end{pmatrix} \longrightarrow \begin{pmatrix} x_{w_{cam}} \\ y_{w_{cam}} \\ z_{w_{cam}} \end{pmatrix} \longrightarrow \begin{pmatrix} X_{u_{cam}} \\ Y_{u_{cam}} \end{pmatrix} \longrightarrow \begin{pmatrix} X_{d_{cam}} \\ Y_{d_{cam}} \end{pmatrix} \longrightarrow \begin{pmatrix} X_{d_{img}} \\ Y_{d_{img}} \end{pmatrix} \tag{2.6}$$

Step 1: This step is the only one regarding the external orientation of the camera. It is defined through the matrix R and the vector T:

$$\begin{pmatrix} x_{w_{cam}} \\ y_{w_{cam}} \\ z_{w_{cam}} \end{pmatrix} = R \cdot \begin{pmatrix} x_{w_{wrl}} \\ y_{w_{wrl}} \\ z_{w_{wrl}} \end{pmatrix} + T \tag{2.7}$$

where R is the 3×3 rotation matrix and T the translation vector:

$$R = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix}, T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{2.8}$$

The 3D rotation represented by the matrix R can be viewed as composition of rotations around the X, Y and Z axis:

$$R = R_X \cdot R_Y \cdot R_Z \tag{2.9}$$

Matrix R_X describing the rotation around the X axis is defined through the angle ψ (roll).

$$R_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{pmatrix} \tag{2.10}$$

Matrix R_Y describing the rotation around the Y axis is defined through the angle ϕ (pitch).

$$R_Y = \begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix} \tag{2.11}$$

Matrix R_Z describing the rotation around the Z axis is defined through the angle θ (yaw).

$$R_Z = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

Multiplication of the matrices R_X, R_Y and R_Z gives us the final form of the rotation matrix R:

$$R = \begin{pmatrix} \cos\phi\cos\theta & -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta & \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ \cos\phi\sin\theta & \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta & -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta \\ -\sin\phi & \sin\psi\cos\phi & \cos\psi\cos\phi \end{pmatrix} \tag{2.13}$$

This means, given the values of the camera's external parameters ψ, ϕ, θ, T_x, T_y and T_z, we can calculate the camera coordinates of the point P_w from Figure 2.2 in the following way (written in homogeneous form):

$$\begin{pmatrix} x_{w_{cam}} \\ y_{w_{cam}} \\ z_{w_{cam}} \\ 1 \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 & T_x \\ r_4 & r_5 & r_6 & T_y \\ r_7 & r_8 & r_9 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{w_{wrl}} \\ y_{w_{wrl}} \\ z_{w_{wrl}} \\ 1 \end{pmatrix} \tag{2.14}$$

where

$$\begin{aligned} r_1 &= \cos\phi\cos\theta \\ r_2 &= -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta \\ r_3 &= \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ r_4 &= \cos\phi\sin\theta \\ r_5 &= \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta \\ r_6 &= -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta \\ r_7 &= -\sin\phi \\ r_8 &= \sin\psi\cos\phi \\ r_9 &= \cos\psi\cos\phi \end{aligned}$$

Step 2: This step is the perspective projection of the point $P_w = (x_{w_{cam}}, y_{w_{cam}}, z_{w_{cam}})$ (Figure 2.2) into the image plane, which gives us the point $P_u = (X_{u_{cam}}, Y_{u_{cam}})$. Given

the focal length f we can apply Equation 2.2:

$$\begin{pmatrix} X_{u_{cam}} \\ Y_{u_{cam}} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{z_{w_{cam}}} & 0 & 0 & 0 \\ 0 & \frac{f}{z_{w_{cam}}} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{w_{cam}} \\ y_{w_{cam}} \\ z_{w_{cam}} \\ 1 \end{pmatrix} \tag{2.15}$$

Step 3: Now we take the lens distortion into account and calculate the camera coordinates of the point $P_d = (X_{d_{cam}}, Y_{d_{cam}})$ based on the camera coordinates of the point $P_u = (X_{u_{cam}}, Y_{u_{cam}})$.

$$\begin{pmatrix} X_{d_{cam}} \\ Y_{d_{cam}} \end{pmatrix} = \begin{pmatrix} X_{u_{cam}} + D_x \\ Y_{u_{cam}} + D_y \end{pmatrix} \tag{2.16}$$

Using Tsai's [Tsa86] lens distortion model D_x and D_y are calculated as follows:

$$\begin{aligned} D_x &= X_{d_{cam}}(\kappa_1 r^2 + \kappa_2 r^4) \\ D_y &= Y_{d_{cam}}(\kappa_1 r^2 + \kappa_2 r^4) \end{aligned} \text{ where } r = \sqrt{X_{d_{cam}}^2 + Y_{d_{cam}}^2} \tag{2.17}$$

κ_1 and κ_2 are the radial and the tangential lens distortion coefficients. Tsai reasons that the tangential distortion (κ_2) can be ignored, which gives us the following equation for calculating the camera coordinates of the distorted point P_d in Figure 2.2:

$$\begin{pmatrix} X_{d_{cam}} \\ Y_{d_{cam}} \end{pmatrix} = \begin{pmatrix} X_{u_{cam}} + \kappa_1 X_{d_{cam}}(X_{d_{cam}}^2 + Y_{d_{cam}}^2) \\ Y_{u_{cam}} + \kappa_1 Y_{d_{cam}}(X_{d_{cam}}^2 + Y_{d_{cam}}^2) \end{pmatrix} \tag{2.18}$$

As one can see, this equation is nonlinear and includes solving cubic equations.

Step 4: In the final step the actual computer image coordinates (in pixels) of the distorted point P_d from Figure 2.2 are calculated. For that we need the internal parameters C_x, C_y, s_x, N_{cx}, N_{fx}, d_x and d_y described in Section 2.2. This is a linear transformation and can be written in homogeneous form:

$$\begin{pmatrix} X_{d_{img}} \\ Y_{d_{img}} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{s_x}{d_{px}} & 0 & C_x \\ 0 & \frac{1}{d_{py}} & C_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} X_{d_{cam}} \\ Y_{d_{cam}} \\ 1 \end{pmatrix} \tag{2.19}$$

where

$$d_{px} = \frac{N_{cx}}{N_{fx}} d_x \text{ and } d_{py} = d_y$$

d_{px} and d_{py} are called *effective* x and y dimension of a pixel in the frame grabber.

2.4 Camera Calibration

Camera calibration is the computation of the camera's internal and external parameters based on a number of points whose world coordinates (x_w, y_w, z_w) are known and image coordinates (X, Y) are measured. There have been many works on camera calibration [AAK71, Tsa86, Rob96, Zha00, JZ01]. They can be roughly divided into two categories,

depending on whether they involve a full-scale nonlinear parameter optimization or perform linear equation solving only. In the methods of the latter category lens distortion can not be considered. In our approach we used the technique proposed by Roger Y. Tsai [Tsa86], which falls into the first category. This method was chosen for several reasons: it is efficient and accurate, lens distortion can be considered but also ignored if desired, and there is a publicly available implementation on Internet [Wil], which has been used for the 3D modeling approach presented.

Tsai's calibration uses the camera model described in Section 2.2, calculating the external and internal camera parameters. Detailed description of Tsai's calibration approach is beyond the scope of this thesis. Interested readers can find the complete description in [Tsa86].

2.5 Plane in 3D Space

There are many ways to represent a plane in 3D space [Wat99]. One of the possibilities is to specify three non-collinear points belonging to the plane (A, B and C in Figure 2.3). If we denote the corresponding vectors of A, B and C with $\vec{a}$, $\vec{b}$ and $\vec{c}$, we can build the

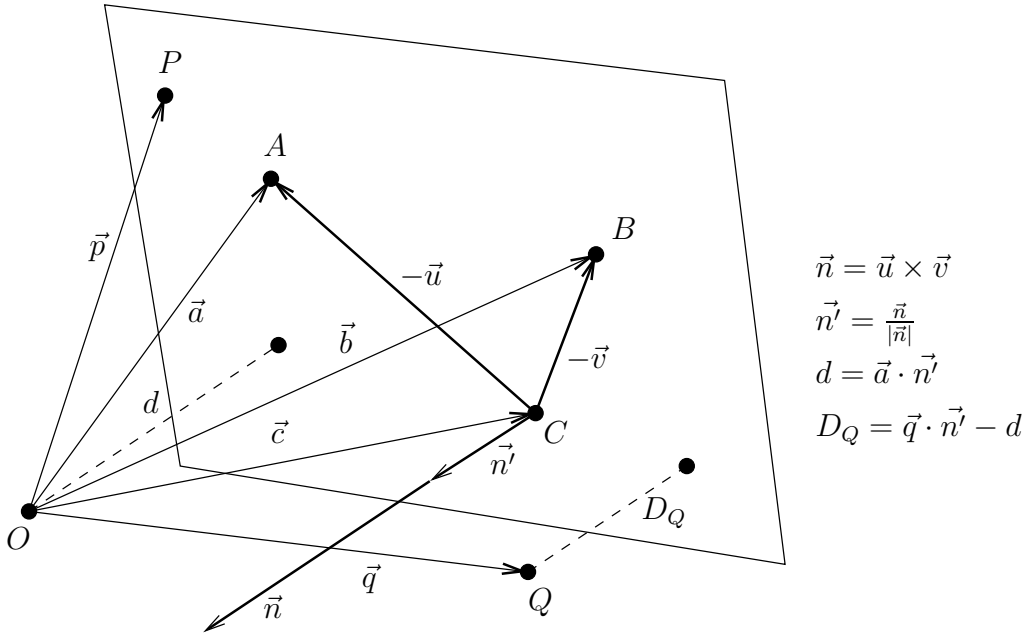


Figure 2.3: Plane in 3D space

vectors $\vec{u}$ and $\vec{v}$ as follows:

$$\begin{aligned} \vec{u} &= \vec{c} - \vec{a} \\ \vec{v} &= \vec{c} - \vec{b} \end{aligned} \tag{2.20}$$

The vectors $\vec{u}$ and $\vec{v}$ are parallel to the plane defined by A, B and C so they can be placed in this plane (see Figure 2.3). The vector product of $\vec{u}$ and $\vec{v}$ is the vector $\vec{n}$ perpendicular

to the plane:

$$\vec{n} = \vec{u} \times \vec{v} \tag{2.21}$$

Dividing this vector with its length $|\vec{n}|$ gives us the corresponding normalized vector $\vec{n'}$ (also illustrated in Figure 2.3):

$$\vec{n'} = \frac{\vec{n}}{|\vec{n}|} \tag{2.22}$$

The vector $\vec{n'}$ is orthogonal to our plane and has the length 1. Building a scalar product of $\vec{n'}$ with any vector representing a point on the plane (e.g., the vector $\vec{a}$ from Figure 2.3) we calculate the distance d between the plane and the origin of the coordinate system:

$$d = \vec{a} \cdot \vec{n'} \tag{2.23}$$

This means, given the plane's normalized orthogonal vector $\vec{n'}$ and the distance d between the plane and the coordinate system's origin, any point P (i.e., vector $\vec{p}$) in the plane satisfies the following equation:

$$\vec{p} \cdot \vec{n'} - d = 0 \tag{2.24}$$

Moreover, the distance D_Q between any point Q (i.e., vector $\vec{q}$) in 3D space and the plane defined through $\vec{n'}$ and d can be calculated as:

$$D_Q = \vec{q} \cdot \vec{n'} - d \tag{2.25}$$

2.6 Summary

This chapter gave an overview of mathematical background necessary to understand the 3D modeling approach presented in this work. The focus was on establishing relations and defining geometrical transformations between different coordinate systems which make a 3D acquisition system — the world, the camera and the image coordinate system. The following chapter describes the acquisition system used for implementation of our 3D modeling method.

Chapter 3

Acquisition System

This chapter presents the acquisition system which has been used for the implementation of the 3D acquisition method proposed in this work. Section 3.1 describes the devices of the acquisition system and their geometrical setup while Section 3.2 defines the relevant coordinate systems. Section 3.3 describes how the calibration points needed for camera calibration are acquired, followed by the description of determining of the laser position in Section 3.4 and the summary in Section 3.5.

3.1 Description

The acquisition system (Figure 3.1) consists of the following devices:

- a turntable (Figure 3.1a) with a diameter of 50 cm, whose desired position can be specified with an accuracy of 0.05° (however, the minimal relative rotation angle is 1.00°). The turntable is used to obtain multiple views of the object observed.
- two monochrome CCD-cameras (Figure 3.1b and 3.1c) with a focal length of 16 mm and a resolution of 768×576 pixels. One camera (Camera-1 in Figure 3.1) is used for acquiring the images of the object's silhouettes and the other (Camera-2 in Figure 3.1) for the acquisition of the images of the laser light projected onto the object.
- a laser (Figure 3.1d) used to project a light plane onto the object. The laser is equipped with a prism in order to span a plane out of the laser beam. The color of the projected light is red.
- a lamp (Figure 3.1e) used to illuminate the scene for the acquisition of the silhouette of the object. The object should be clearly distinguishable from the background. This can be best achieved with backlighting [HS91].

The geometrical setup of the acquisition devices is shown in Figure 3.2. Both cameras are placed in a distance of about 50 cm from the rotational axis of the turntable. Ideally the optical axis of the camera for acquiring object's silhouettes (Camera-1 in Figure 3.2) lies nearly in the rotational plane of the turntable, orthogonal to the rotational axis. The

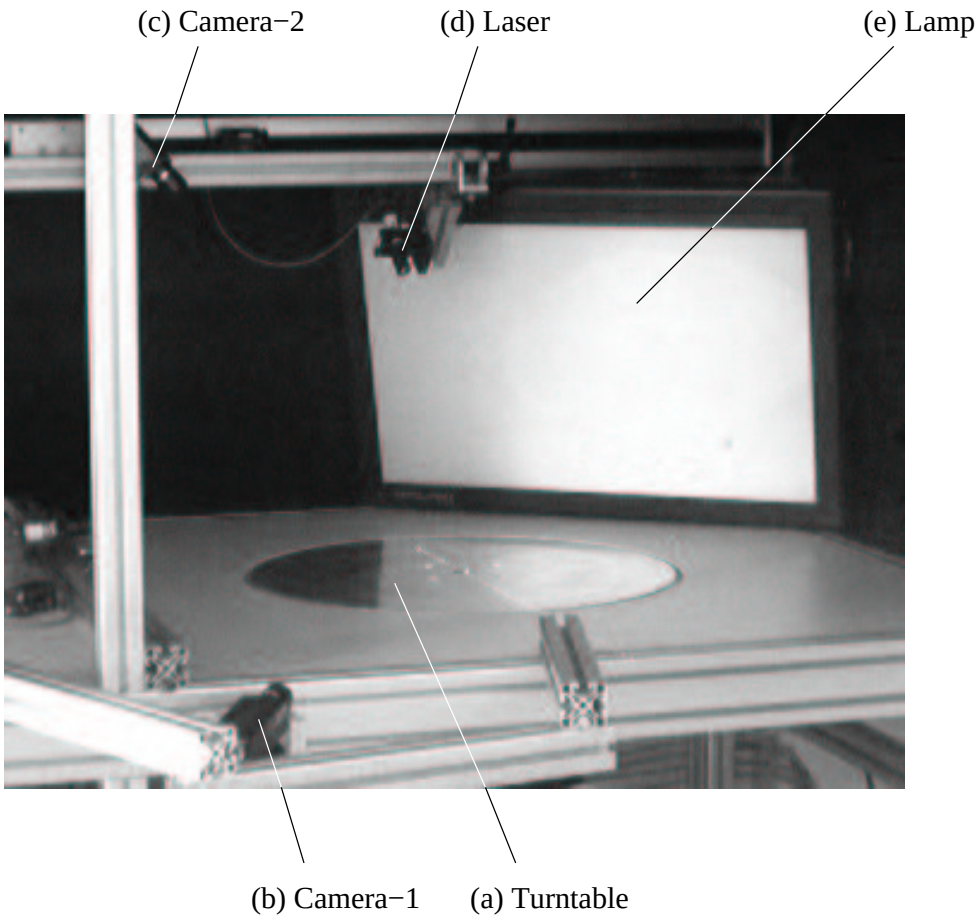


Figure 3.1: Acquisition System

camera for acquiring the projection of the laser plane (Camera-2 in Figure 3.2) onto the object views the turntable from an angle of about 45° (β in Figure 3.2). The laser is directed such that the light plane it projects contains the rotational axis of the turntable. Camera-2 from Figure 3.2 views the light plane also from an angle of about 45° (α in Figure 3.2). The relative position of the two cameras to one another is not important, since the acquisition of the silhouettes and the acquisition of the laser light projection are independent from one another.

3.2 Relevant Coordinate Systems

Before 3D reconstruction of any object can take place, the acquisition system has to be calibrated, i.e., the relative positions of the cameras, the lasers and the turntable have to be computed.

For the acquisition system described in this section we define the following coordinate systems (Figure 3.3), all of which are right-handed:

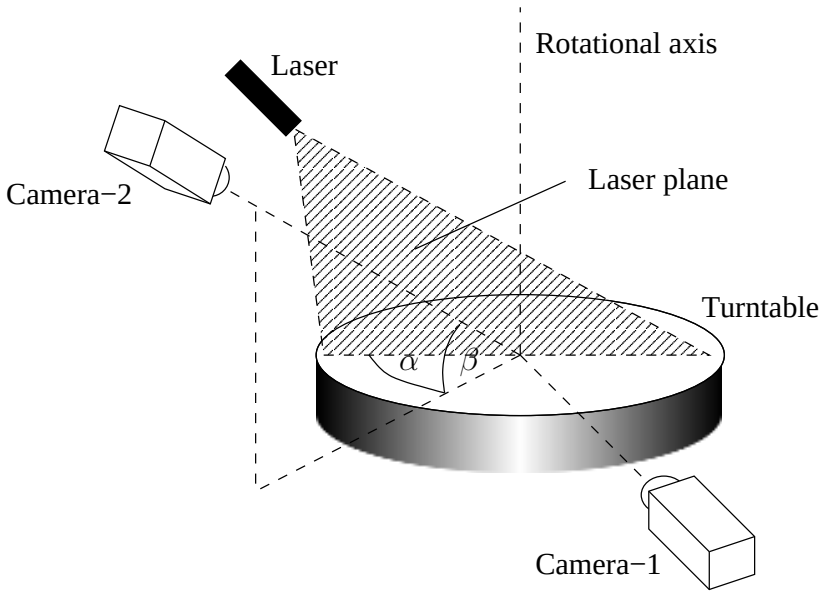


Figure 3.2: Geometrical setup of acquisition system

- *world* coordinate system: it is rooted at the intersection of the rotational axis of the turntable and its rotational plane. Its z axis is identical to the rotational axis of the turntable. x and y axes lie in a plane parallel to the turntable's rotational plane. The x axis is positioned such that the x-z plane is identical to the laser plane.

- *camera-1* coordinate system: rooted at the center of the lens of the Camera-1 from Figure 3.2, with the z axis being identical to the camera's optical axis, and x and y axes as shown in Figure 3.3.

- *camera-2* coordinate system: rooted at the center of the lens of the Camera-2 from Figure 3.2, with z axis being identical to the camera's optical axis, and x and y axes as shown in Figure 3.3.

- *image-1* coordinate system: 2D image coordinate system for images taken with Camera-1.

- *image-2* coordinate system: 2D image coordinate system for images taken with Camera-2.

With this definition of the relevant coordinate systems the calibration has to provide the following information:

1. the transformation from world to image-1 coordinate system, which is a composition of transformations from world to camera-1 and from camera-1 to image-1 coordinate systems.

2. the transformation from world to image-2 coordinate system, which is a composition of transformations from world to camera-2 and from camera-2 to image-2 coordinate systems.

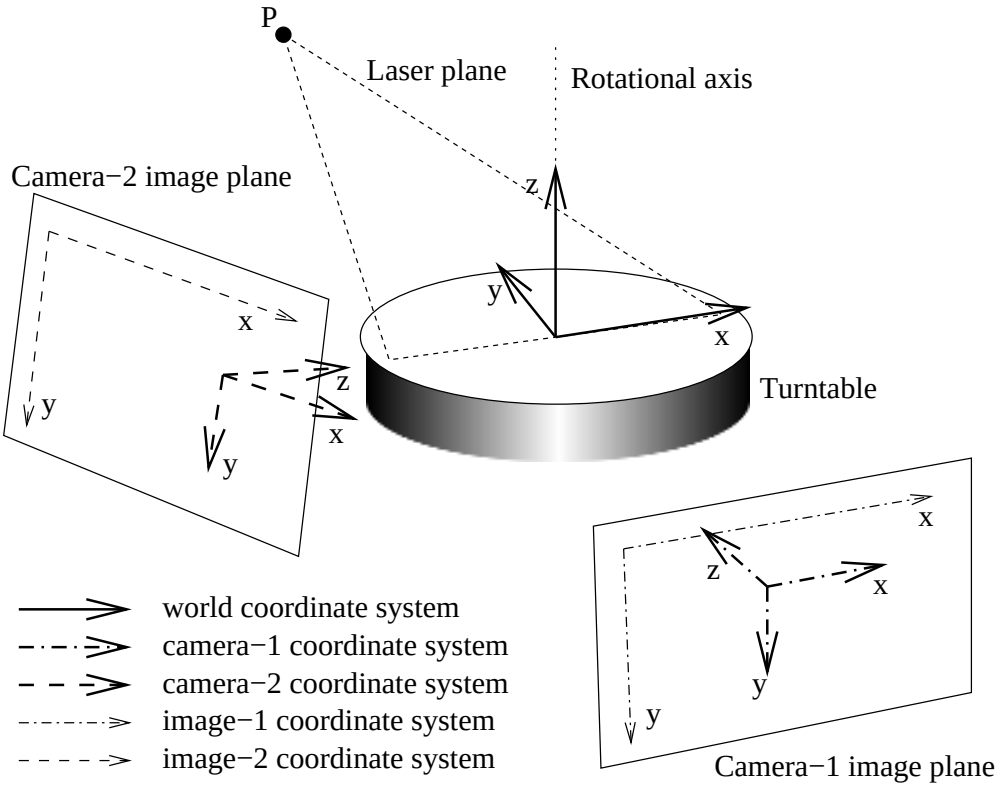


Figure 3.3: Relevant coordinate systems

3. the position of the laser in world coordinate system, i.e., the world coordinates of the point source of the laser plane (point P in Figure 3.3).

The first two transformations and the parameters needed to perform them were described in Section 2.3. Section 2.4 gave a general overview of camera calibration. What still has not been described is how the pairs of 3D world calibration points and their corresponding 2D image points are obtained.

3.3 Acquisition of Calibration Points

As noted in Section 2.4, in order to perform camera calibration, a set of 3D points is needed whose world coordinates are known and whose 2D image coordinates are measured. Tsai's calibration method [Tsa86] needs at least 7 pairs of 3D world and the corresponding 2D image points. However, for fully optimized calibration at least 11 points are needed. The 3D points can be either coplanar or non-coplanar. The calibration is more accurate with non-coplanar 3D data, because with planar data the internal parameter s_x can not be optimized. The accuracy of the calibration increases with the increasing number of points.

In our approach we use a pattern of 25 circles in 5 rows and 5 columns, with the distance of 30 mm between the centers of two neighboring circles in each row/column (Figure 3.4).

The circles were made large enough to contain a sufficient number of pixels in an image in order to calculate their centers with a sub-pixel accuracy, and small enough that the coordinates of the centers can be approximated by the mean value of the coordinates of all points belonging to the corresponding circle.

In order to obtain non-coplanar calibration points, in our experiments the pattern was

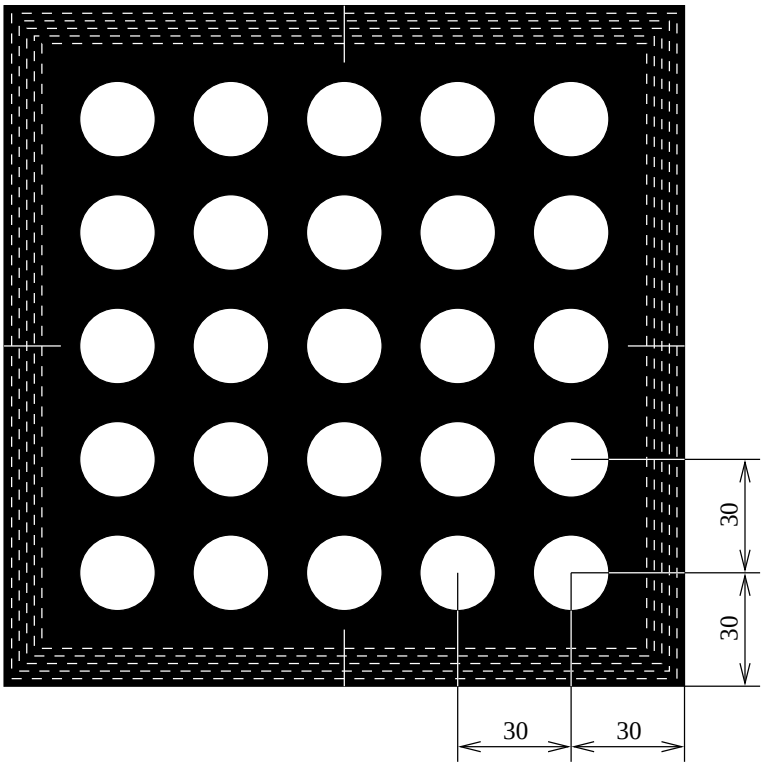


Figure 3.4: Calibration Pattern

placed in 3 different planes of the world coordinate system, thus providing us with up to 75 points defined by the circle centers.

As shown in Figure 3.3, we make the z axis of the world coordinate system identical to the rotational axis of the turntable and the x-z plane identical to the laser light plane. The aligning of the laser plane with the x-z plane simplifies processing of laser light images. In order to place the calibration pattern from Figure 3.4 "correctly" in the world coordinate system, we perform three steps:

Step 1: A metal rod in placed the center of the turntable (Figure 3.5a). Then two lasers (one of which we will use for the Shape from Structured Light part of the method proposed) are oriented such that the two light planes are perpendicular to one another, intersecting in the turntable's rotational axis approximated by the rod (Figure 3.5b).

Step 2: The rod is taken out and a piece of paper containing a 2D coordinate system is placed on the turntable's surface such that the planes of the two lasers which were oriented on the rod in Step 1 pass through the axes on the paper (Figure 3.6). Doing so,

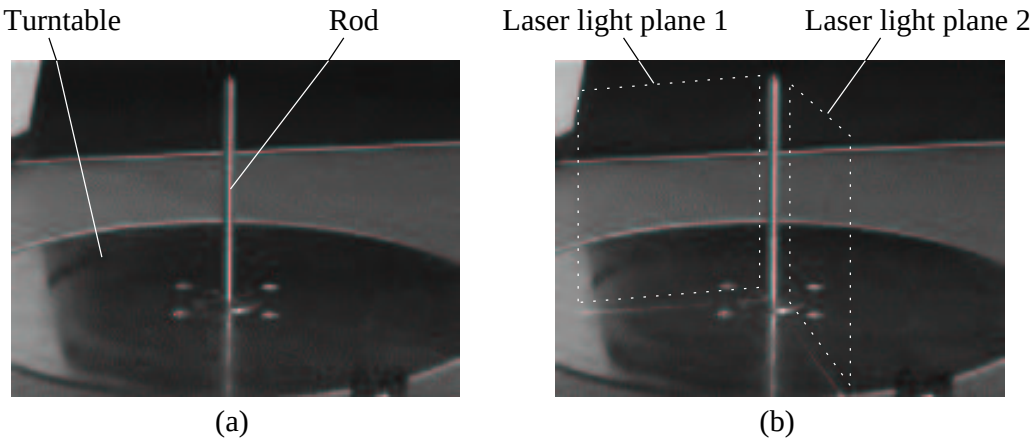


Figure 3.5: Aligning lasers on turntable's rotational axis

we align the z axis of the world coordinate system with the turntable's rotational axis. The x axis is set to lie in the light plane of the laser used later for data acquisition.

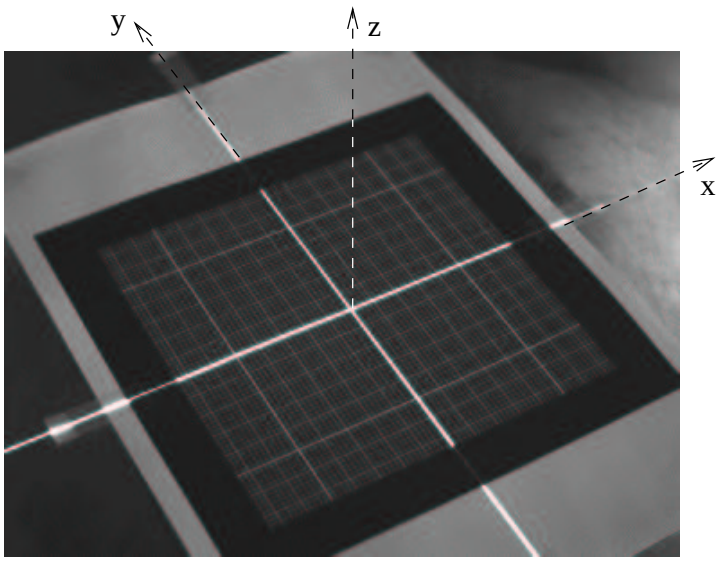
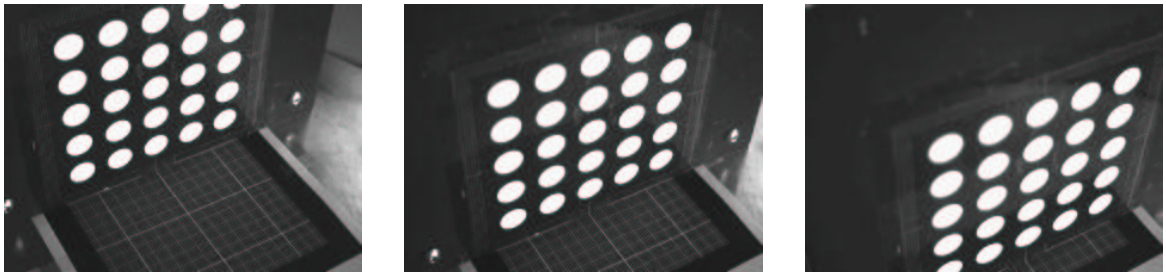


Figure 3.6: Setting up the world coordinate system

Step 3: The calibration pattern from Figure 3.4 is placed on any desired position on the turntable, vertically to the world x-y plane. In our experiments we placed it in 3 planes parallel to the world x-z plane — one plane with $y = 50$ (Figure 3.7a), one with $y = 0$ (Figure 3.7b) and one with $y = -50$ (Figure 3.7c). Doing so, the calibration points are spread across the acquisition space where the objects are placed.

Knowing the dimensions of the calibration pattern from Figure 3.4, after performing these 3 steps we know the position of the pattern in our world coordinate system and



(a) (b) (c)

Figure 3.7: Placement of the calibration pattern

therefore we know the world coordinates of the circle centers. Now we need to measure their corresponding image coordinates. This can be done by finding connected components in the image (in this case circles) and calculating the mean image coordinates of each component, illustrated in Figure 3.8.

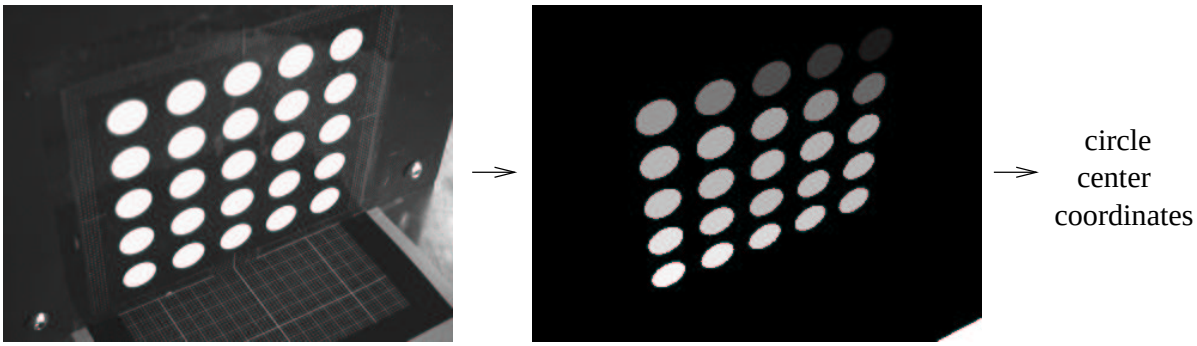


Figure 3.8: Calculating coordinates of circle centers

Another possibility for calculation of circle centers using the same calibration pattern is to detect ellipses in the images, estimate their principal axes, and calculate the centers by intersecting the axes of the corresponding ellipse. This method is computationally more complex, but it could achieve higher accuracy.

The pairs of 3D world coordinates of circle centers and their corresponding 2D image coordinates acquired as described above are used as input for camera calibration (see Section 2.4).

3.4 Calculation of the Laser Position

The position of the laser is calculated by intersecting two rays coming from the laser. The rays are obtained using an object with known dimensions and placing it in the laser plane, i.e., in the x-z plane of the world coordinate system. We used a metal cuboid shown in Figure 3.9. Using a grid paper representing the world x-y plane and knowing the height

of the cuboid we can read the coordinates of two points belonging to the ray (P_1 and P_2 in Figure 3.9). Placing the cuboid in two different positions, the world coordinates of 4

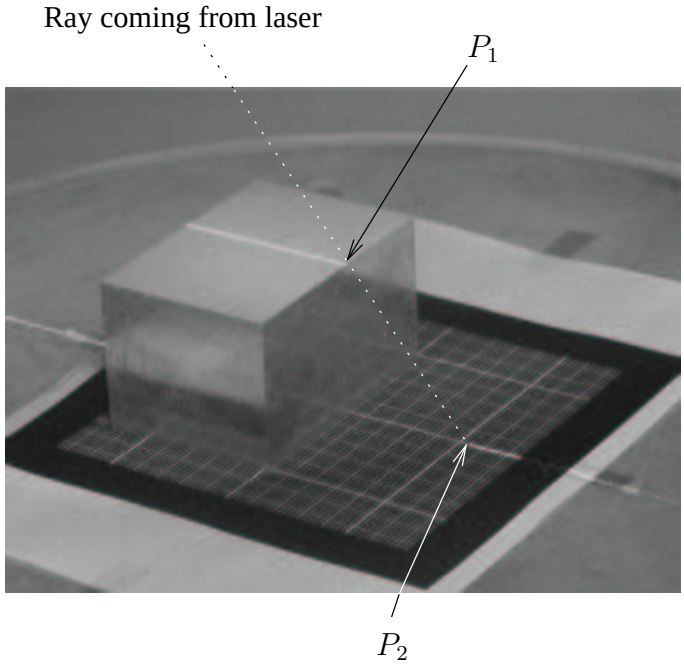


Figure 3.9: Cuboid in laser plane

points P_{11}, P_{12}, P_{21} and P_{22}, can be read, defining two rays, as illustrated in Figure 3.10. The intersection of the rays defined by $\overline{P_{11}P_{12}}$ and $\overline{P_{21}P_{22}}$ gives the position of the laser,

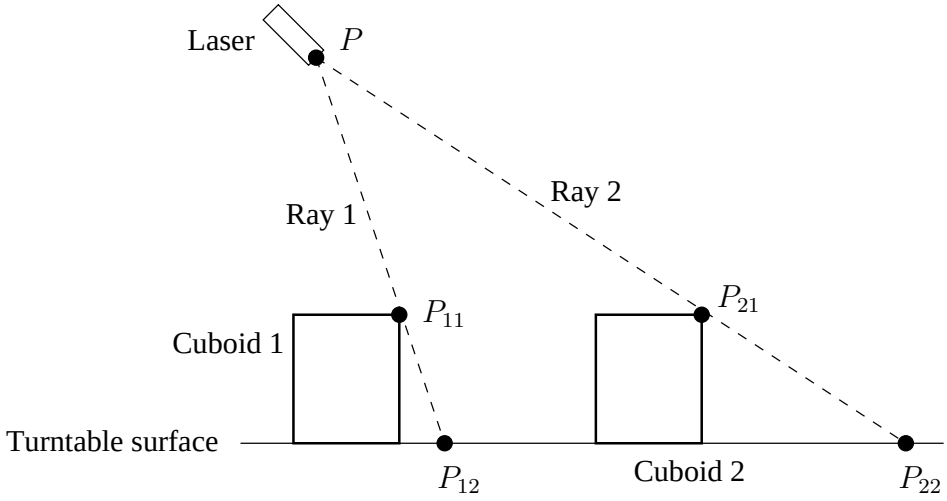


Figure 3.10: Intersecting laser rays

point P in Figure 3.10. If we denote the coordinates of the point P_{ij} with (x_{ij}, y_{ij}, z_{ij}), then the coordinates (x_P, y_P, z_P) of the point P, with the assumption that $y_{ij} = 0$ for

every i, j, are calculated as follows:

$$\begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix} = \begin{pmatrix} x_{11} \\ y_{11} \\ z_{11} \end{pmatrix} + t \cdot \begin{pmatrix} x_{12} - x_{11} \\ y_{12} - y_{11} \\ z_{12} - z_{11} \end{pmatrix} \tag{3.1}$$

where

$$t = \frac{(x_{11} - x_{21})(z_{22} - z_{21}) - (x_{22} - x_{21})(z_{11} - z_{21})}{(x_{22} - x_{21})(z_{12} - z_{11}) - (x_{12} - x_{11})(z_{22} - z_{21})}$$

3.5 Summary

This chapter presented the acquisition system used for design and implementation of the 3D modeling method presented. It described its components — the cameras, the laser, the turntable — and how the relative positions of these components are determined, i.e., how the system is calibrated. The calibration of the acquisition system is a prerequisite for any 3D modeling approach. Possible approaches are discussed in the following chapter.

Chapter 4

3D Modeling — Possible Approaches

This chapter addresses the questions that need to be answered when designing an algorithm for building a 3D model of an object, with focus on combining Shape from Silhouette and Shape from Structured Light. Section 4.1 gives an overview of types of 3D model representation, followed by a more detailed description of the octree representation in Section 4.1.3. Section 4.2 describes the main characteristics of the Shape from Silhouette and Shape from Structured Light based models and problems encountered when trying to combine them. Section 4.3 concludes the discussion of possible approaches, leading to the approach selected.

4.1 Types of 3D Model Representation

There are many different model representations in computer vision and computer graphics used. Most of them can be roughly categorized into *surface*-based and *volume*-based representations.

4.1.1 Surface-Based Representations

Surface-based representations describe a three-dimensional object by defining the surfaces that bound the object. The bounding surfaces can be represented by different computer graphics methods. For example, they can be represented by a cloud of points belonging to the surface (an amphora in Figure 4.1a) or as a set of simple approximating patches, like planar or quadric patches [BJ88]. Figure 4.1b shows a sphere represented by quadrilateral planar patches. Points and patches representing the surface can also be associated with a normal vector, which is the vector of the length 1 and normal to the associated point or patch. An advantage of this kind of representation is that it can be used for arbitrarily shaped objects and it is particularly useful for object visualization. However, it is difficult to perform any 3D measurements on the model, such as volume computation.

Another surface-based representation, usually used for solid objects, is the B-Rep [Wat99] representation. It describes an object as a volume enclosed by a set of primitive surface elements, typically sections of planes and quadratic surfaces such as spheres,

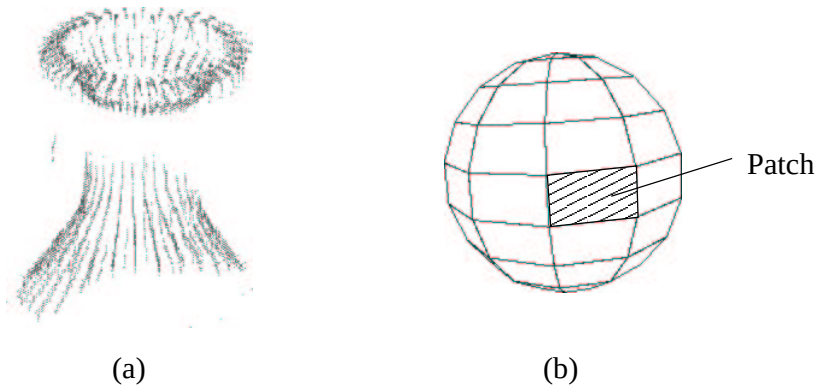


Figure 4.1: Cloud of points (a) and planar patches (b)

cylinders and cones. This representation is shown in Figure 4.2. The volume of a B-Rep

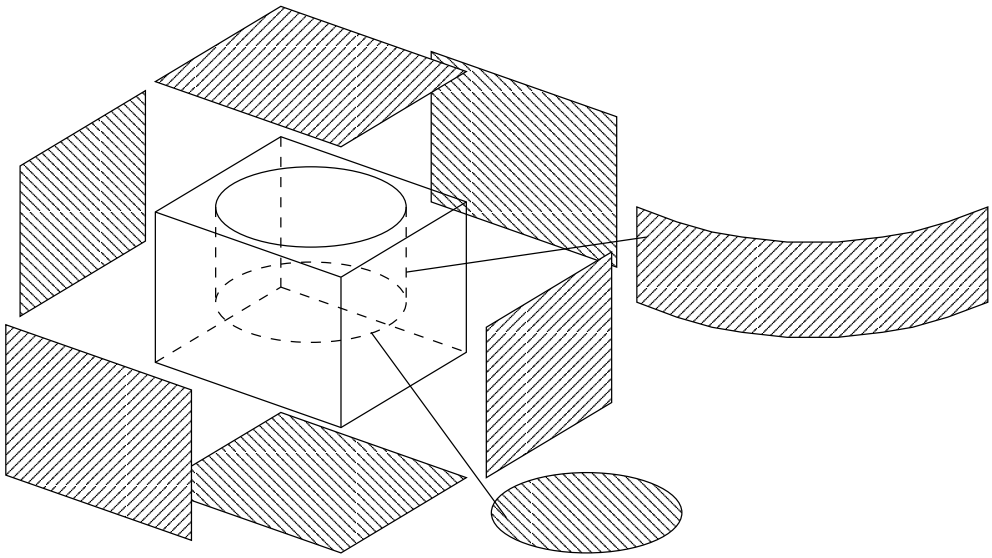


Figure 4.2: B-Rep representation

model of an object can be computed analytically, but the complexity of the model grows quadratically as the complexity of the object increases.

Generalized cylinder [MN78] representation describes the surface of an object by a two-dimensional cross-sectional figure which is swept along a three-dimensional space curve acting as the axis (also called spine) of the cylinder. The cross section can vary smoothly along the axis. Each cross section is orthogonal to the cylinder axis. This representation is illustrated in Figure 4.3. With generalized cylinder representation of an object the computation of the volume is straight-forward — the integral of the surface of the cross-sectional figure is built along the cylinder axis. However, this representation is suitable for a specific set of shapes only and it is not flexible enough to describe arbitrary solid objects.

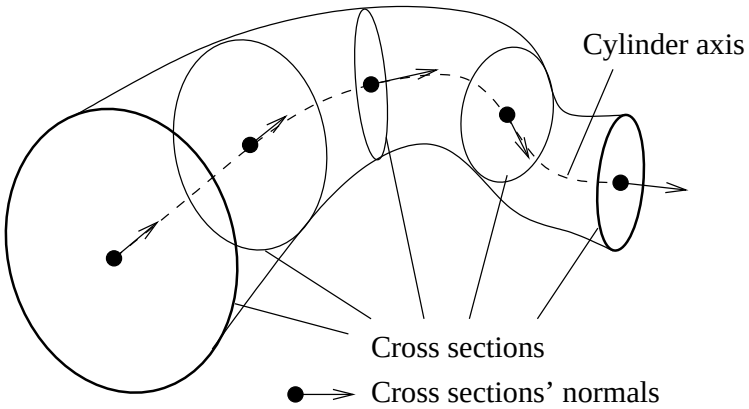


Figure 4.3: Generalized cylinder representation

4.1.2 Volume-Based Representations

Volume-based representations use volume elements rather than surface elements to describe an object. Constructive Solid Geometry (CSG) [Wat99, Shi87] method uses simple volumetric primitives, such as blocks, cones, cylinders and spheres, and a set of boolean operations — union, intersection, and difference. Figure 4.4 shows a CSG representation of a simple object. While the computation of the volume of an object based on a CSG model is relatively simple, this representation is not suitable for arbitrarily curved objects.

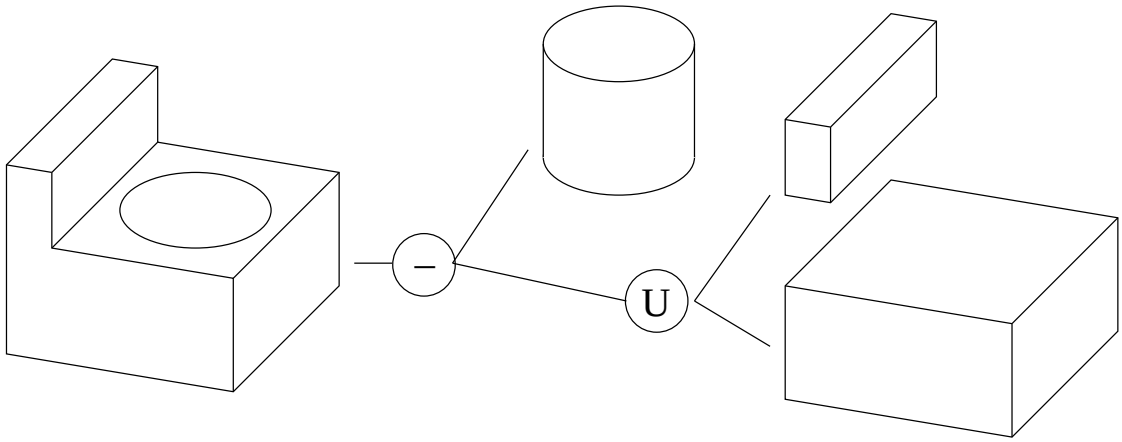


Figure 4.4: Constructive Solid Geometry (CSG)

Another kind of volume-based representations are the so called spatial occupancy representations. An object is represented using non-overlapping subregions of the 3D space occupied by the object. There are many variants of this representation, such as voxel (volume element) or octree [CH88] representation. In a voxel representation, an

object is built up from a 3D binary array, where elements have the value 1 if occupied, otherwise 0. The resolution is uniform throughout the representation. Figure 4.5 shows a voxel representation of an object. Octree representation is just an efficient way of representing 3D voxel array describing an object. The Octree is the representation we chose for the 3D modeling approach presented in this work and it is described in detail in the following section.

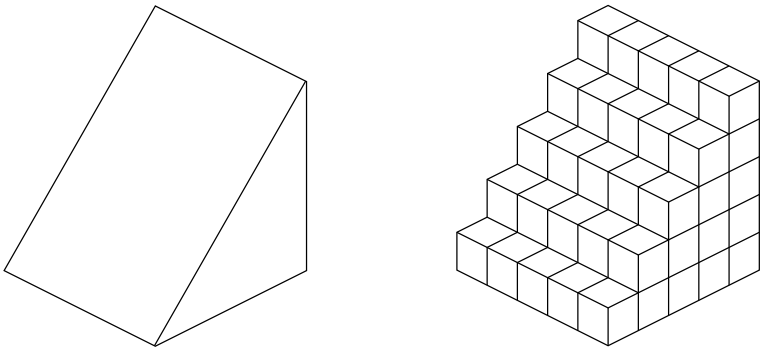


Figure 4.5: An object (a) and its voxel representation (b)

Spatial occupancy representations can be used for arbitrarily shaped objects, but they give a low-level description of an object which is unsuitable for some applications, such as object recognition. However, it is suitable for volume computation and other 3D measurements of arbitrarily shaped objects.

4.1.3 Octree Model Representation

An octree [CH88] is a tree-formed data structure used to represent 3-dimensional objects. Each node of an octree represents a cube subset of a 3-dimensional volume. A node of an octree which represents a 3D object is said to be:

- *black*, if the corresponding cube lies completely within the object
- *white*, if the corresponding cube lies completely within the background, i.e., has no intersection with the object
- *gray*, if the corresponding cube is a boundary cube, i.e., belongs partly to the object and partly to the background. In this case the node is divided into 8 child nodes (octants) representing 8 equally sized subcubes of the original cube

All leaf nodes are either black or white and all intermediate nodes are gray. An example of a simple 3D object and the corresponding octree is shown in Figure 4.6.

An octree as described above contains binary information in the leaf nodes and therefore it is called a binary octree, and it is suitable for representation of 3D objects where

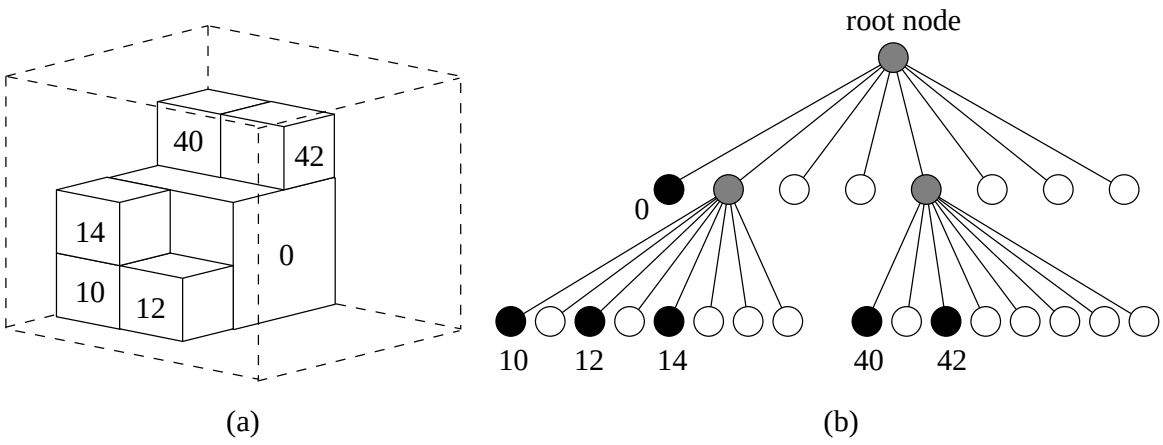


Figure 4.6: A simple object (a) and the corresponding octree (b)

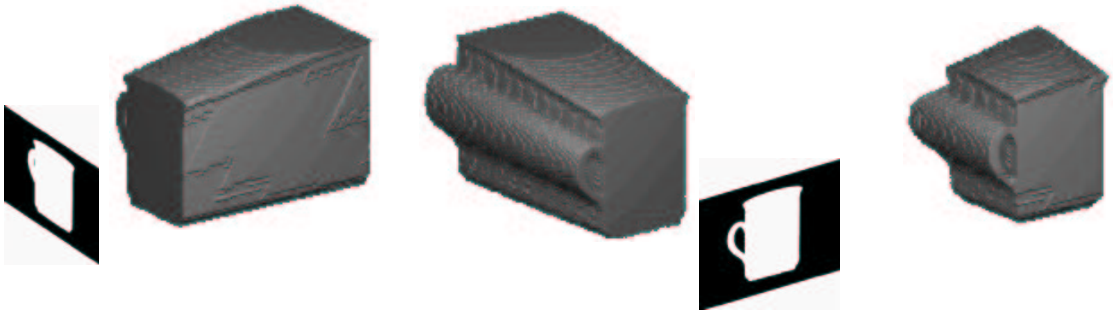
the shape of the object is the only object property that needs to be modeled by the octree. Non-binary octrees can contain other information in the leaf nodes, e.g., the cube color in RGB-space. For the 3D modeling approach presented in this work, a binary octree model is sufficient to represent 3D objects.

Octree model representation has several advantages: for a typical solid object it is an efficient representation, because of a large degree of coherence between neighboring volume elements (voxels), which means that large pieces of an object can be represented by a single octree node. Another advantage is the ease of performing geometrical transformations on a node, because they only need to be performed on the node's vertices. The disadvantage of octree models is that they digitize the space by representing it through cubes whose resolution depend on the maximal octree depth and therefore cannot have smooth surfaces. However, this is a problem with any kind of voxel-based volumetric representation.

4.2 Building a 3D Model

An input image for *Shape from Silhouette* defines a conic volume in space which contains the object to be modeled (Figure 4.7a). Another input image taken from a different view defines another conic volume containing the object (Figure 4.7b). Intersection of the two conic volumes narrows down the space the object can possibly occupy (Figure 4.7c). With an increasing number of views the intersection of all conic volumes approximates the actual volume occupied by the object better and better, converging to the 3D convex hull of the object. Therefore by its nature *Shape from Silhouette* defines a *volumetric* model of an object.

An input image for *Shape from Structured Light* using laser light defines solely the points on the surface of the object which intersect the laser plane (Figure 4.8a). Using multiple views provides us with a cloud of points belonging to the object surface (Figure 4.8b), i.e., with the *surface* model of the object.



(a) (b) (c)

Figure 4.7: Two conic volumes and their intersection

(a) (b)

Figure 4.8: Laser projection and cloud of points

The main problem that needs to be addressed in an attempt to combine these two methods is how to adapt the two representations to one another, i.e., how to build a common 3D model representation. This can be done in several ways:

- Building the *Shape from Silhouette*'s volumetric model and the *Shape from Structured Light*'s surface model independently from one another. Then, either convert the volumetric model to a surface model and use the intersection of the two surface models as the final representation or convert the surface model to a volumetric model and use the intersection of the two volumetric models as the final representation.

- Using a common 3D model representation from the ground up, avoiding any model conversions. That means either design a volume based Shape from Structured Light algorithm or a surface based Shape from Silhouette algorithm.

With the former method both underlying algorithms would build their "native" model of the object. However, conversion and intersection of the models would not be a simple task. While conversion of the Shape from Silhouette's volumetric model to a surface model is straightforward — one only has to find 3D points of the volume belonging to the surface — an intersection of two surface models can be rather complex. One could

start from the points obtained by Shape from Structured Light (because they really lie on the object's surface, whereas points on the surface of the volume obtained by Shape from Silhouette only lie somewhere on the object's convex hull) and fill up the missing surface points with points from the Shape from Silhouette model. There are several problems with this approach. There could be many "jumps" on the object surface, because the points taken from the Shape from Silhouette model might be relatively far away from the actual surface. The approach would also not be very efficient, because we would need to build a complete volumetric model through Shape from Silhouette, then intersect it with every laser plane used for Shape from Structured Light in order to create a surface model, and then, if we also want to compute the volume of the object, we would have to convert the final surface model back to the volumetric model.

Another possibility would be converting the surface model obtained by Shape from Structured Light to a volumetric model and intersect it with the Shape from Silhouette's model. In this case the intersection is the easier part — for each voxel of the space observed one would only have to look up whether both models "agree" that the voxel belongs to the object — only such voxels would be kept in the final model and all others defined as background. Also the volume computation is simple in this case — it is a multiplication of the number of voxels in the final model with the volume of a single voxel. But the problem with this approach is the conversion of the Shape from Structured Light's surface model to a volumetric model — in most cases, the surface model obtained using laser plane is very incomplete (see the model of an amphora in Figure 4.8b) because of the light and camera occlusions (Figure 4.9), so one would have to decide how to handle the missing parts of the surface. Also, the conversion of a surface model to a volumetric

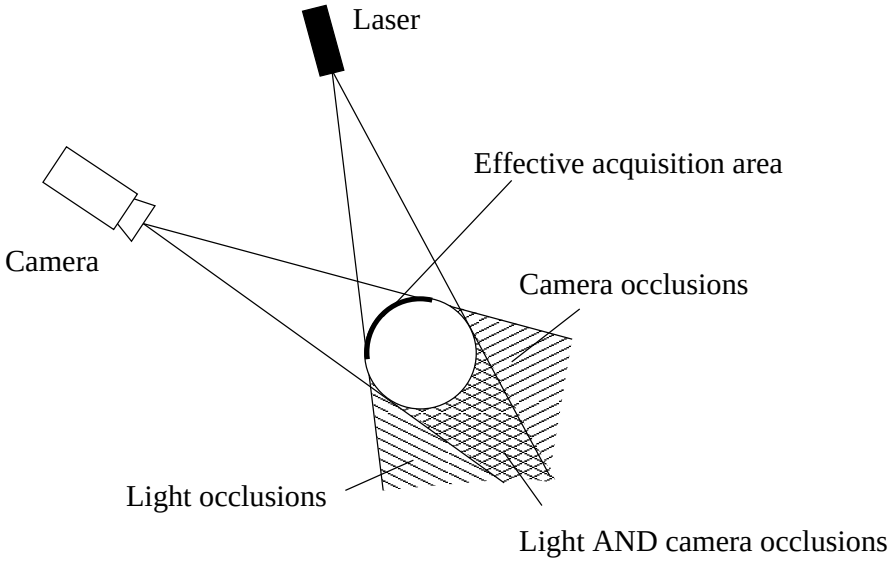


Figure 4.9: Light and camera occlusions

model is generally a complex task, because if the surface is not completely closed, it is hard to say whether a certain voxel lies inside or outside the object. With closed surfaces

one could follow a line in 3D space starting from the voxel observed and going in any direction and count how many times the line intersects the surface. For an odd number of intersections one can say that the voxel belongs to the object. But even in this case there would be many special cases to handle, e.g., when the chosen line is tangential to the object's surface.

4.3 Conclusions on Possible Approaches

The discussion of possible approaches in Section 4.2 lead us to the following conclusions:

- Building a separate Shape from Structured Light surface model and a Shape from Silhouette volumetric model followed by converting one model to the other and intersecting them is mathematically complex and computationally costly.
- If we want to estimate the volume of an object using our model, any intermediate surface models should be avoided because of the problems of conversion to a volumetric model.
- The modeling approach presented in this work will be mostly applied to archaeological vessels and sherds, which are often objects with many differently sized and shaped curvatures, so we need a model representation which is flexible enough to represent arbitrarily shaped objects and makes it easy to calculate the volume of an object.

Therefore, our approach proposes building a single octree model from the ground up, using both underlying methods in each step.

When building a 3D volumetric model of an object based on a number of its 2D images, there are two possibilities regarding the decision whether a certain voxel is a part of the object or belongs to the background. One possibility is to start from the input images and project all pixels belonging to the object into the 3D space, finding all the voxels which can possibly be occupied by the object (for example, that would be the conic volume corresponding to the image of a coffee mug in Figure 4.7a). Then, these voxels can be intersected with the voxels defined through the other input images (Figure 4.7b) in order to obtain the final model (Figure 4.7c). Another possibility is to start from a limited 3D space and project its voxels into all input images, testing whether all input images "agree" that the voxel belongs to the object. This is illustrated in Figure 4.10. Each voxel of the starting object space is projected into every input image and intersected with the object's image representation. The former approach requires intersection testing in 3D space whereas the latter approach requires intersection testing in 2D image planes. Therefore, we chose to use the latter approach, i.e., project the 3D space into the input images.

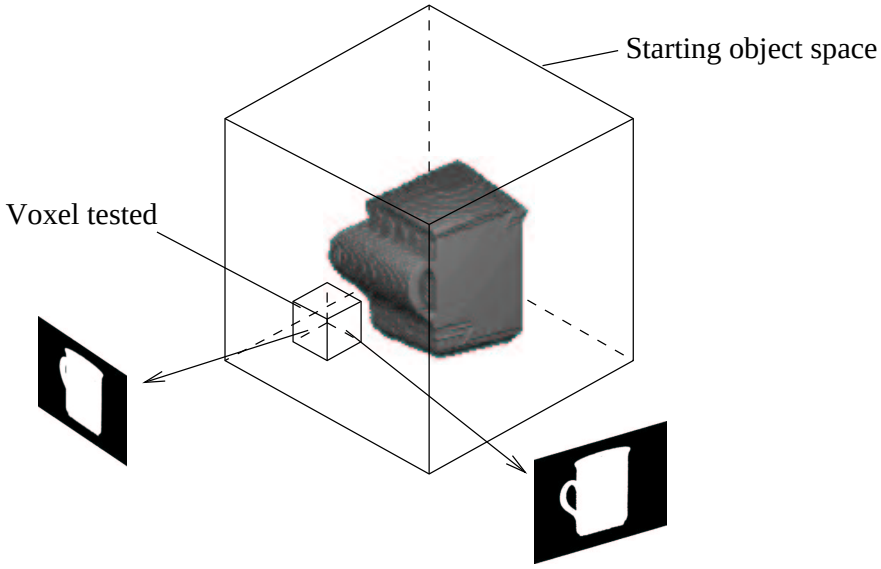


Figure 4.10: Voxel occupancy test

Chapter 5

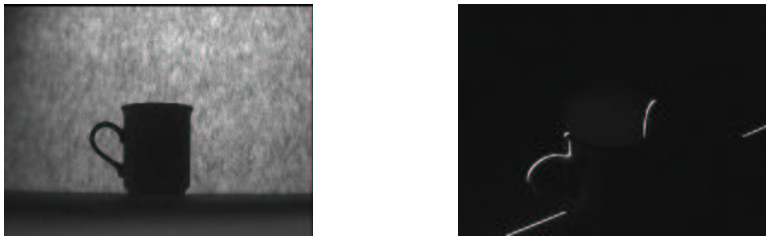
3D Modeling — Octree Based Approach

This chapter presents our 3D modeling approach in detail. It assumes having a fully calibrated acquisition system and describes what happens between acquiring a set of silhouette and laser light images of an object from different views and producing the final 3D model of the object. In short, our approach takes the images acquired, binarizes them and builds an octree model of the object by projecting the octree nodes into the relevant binary images and intersecting them with the object's image representation, in order to decide whether the node belongs to the object or to the background. The following sections go into details of each of these steps. Section 5.1 describes the acquisition of silhouette and laser light images of an object and Section 5.2 the binarization of the images acquired. Section 5.3 presents the steps of projection of an octree node into both kinds of input images, followed by the description of the intersection test of a node with the object's image representation in Section 5.4. Finally, Section 5.5 puts all the pieces together, giving a step-by-step description of the model building algorithm, followed by the summary in Section 5.6.

5.1 Image Acquisition

The acquisition system with all its components was described in detail in Chapter 3 (see also Figure 3.1). For image acquisition, an object is placed on the turntable (Figure 3.1a), approximately in its center, so that the object stays close to the center of the images acquired, independent from the rotation angle of the turntable. During the acquisition process, the whole acquisition space is protected against ambient light by a thick black curtain.

For the acquisition of silhouette images, Camera-1 from Figure 3.1 is used, and the lamp (Figure 3.1e) is switched on for backlighting of the object, in order to create high contrast between the object and the background. As an example, a silhouette image of a coffee mug is shown in Figure 5.1a.



(a) (b)

Figure 5.1: Sample silhouette (a) and laser light (b) image of a coffee mug

For the acquisition of laser light images, Camera-2 (Figure 3.1c) and the laser (Figure 3.1d) are used. Figure 5.1b shows a laser light image of a coffee mug.

In order to acquire multiple views of an object, the turntable is either rotated by a fixed angle between two neighboring views (usually in the range 1°–20°, resulting in 18–360 views) or this angle is computed automatically for each view. The latter method is covered in detail in Chapter 6, Next View Planning.

5.2 Binarization of the Acquired Images

The first step between the image acquisition and creation of the final 3D model of an object consists of converting the images acquired into binary images. A pixel in such a binary image should have the value 0 if it represents a point in 3D space which does not belong to the object *for sure*, and the value of 1 otherwise (i.e., if it represents a point *possibly* belonging to the object). The purpose of having such binary images is to simplify the processing of images during the model building process — a simple lookup of the value of a pixel (0 or 1) tells whether the pixel represents the background or the object. The binarization is performed on input images for both Shape from Silhouette and Shape from Structured Light.

5.2.1 Binarization of Shape from Silhouette Images

For the Shape from Silhouette part of the method presented in this work, a reliable extraction of the object's silhouette from an acquired image is of crucial importance for obtaining an accurate 3D model of an object. If the background brightness is not uniform, i.e., if there are parts of the background which are brighter than others (especially if there are parts with similar brightness like the object observed), the silhouette extraction can be a difficult task. For that reason, in addition to the images of the object (Figure 5.2a) taken from different viewpoints, an image of the acquisition space (Figure 5.2b) is taken, without any object in it. Then, the absolute difference between this image and an input image is built, which creates an image (Figure 5.2c) with a uniform background and a high contrast between the object and the background. Then, thresholding is used to create a binary image (Figure 5.2d) where pixels with the value 1 represent the object's

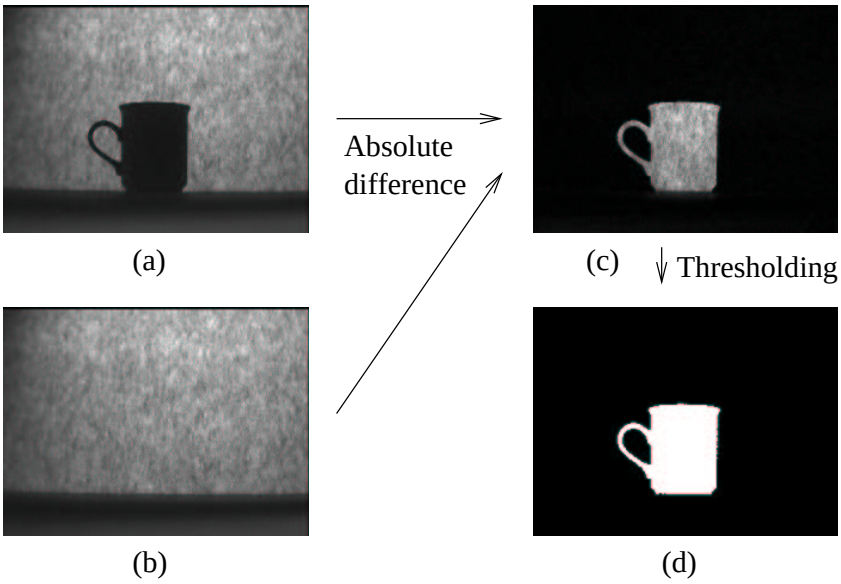


Figure 5.2: Binarization of Shape from Silhouette images

silhouette and those with value 0 the background. The threshold value is either calculated automatically or specified by the user.

Another option for extracting object silhouettes from input images would be to use edge detection [NB86] instead of thresholding. This approach could be more accurate, even a sub-pixel precision could be reached, but it is also more complex. One of the questions that would have to be answered, given the edges of the object, is how to decide what is inside and what outside the object. This question might be easy to answer for simple objects, but for more complex objects (e.g. archaeological vessels with a thin handle) it could be rather complicated.

By using backlighting for acquisition of silhouette images, saturation problems could occur — the brighter image areas tend to be larger than they are, what could add some error to the silhouette extraction. Because of this it is important to establish the lowest brightness possible, just sufficient to distinguish the object from the background.

5.2.2 Binarization of Shape from Structured Light Images

An input image for Shape from Structured Light contains the projection of a laser plane onto the object (Figure 5.3). A white pixel in this image represents a 3D point on the object's surface which intersects the laser plane. A black pixel represents a 3D point in the laser plane which does not belong to the object's surface — it is either inside the object or it does not belong to the object at all. For our approach, if we know that a pixel represents a point outside the object, we want to leave it black. If we know it is inside the object or if we are not sure, we want to paint it white. How can we decide whether a black pixel corresponds to a point inside or outside the object? If we know where the

laser light is coming from we can project it into the image plane (point P in Figure 5.3) and draw a line through this point and the black pixel we want to test (points Q_1, Q_2 and Q_3 in Figure 5.3). If there is a white pixel R (i.e., a pixel belonging to the laser line) between P and the pixel tested, like Q_1 in Figure 5.3, then Q_1 can not be reached by the laser light and therefore we can not say whether it is inside or outside the object and it should be changed to white. Q_2 from Figure 5.3 lies between P and R so it is for sure outside the object and should remain black. Finally, if there is no white pixel on the line between P and the pixel tested, such as Q_3 in Figure 5.3, this pixel is occluded by a part of the object outside the laser plane so we can not tell anything about it being inside or outside the object and we set it to white.

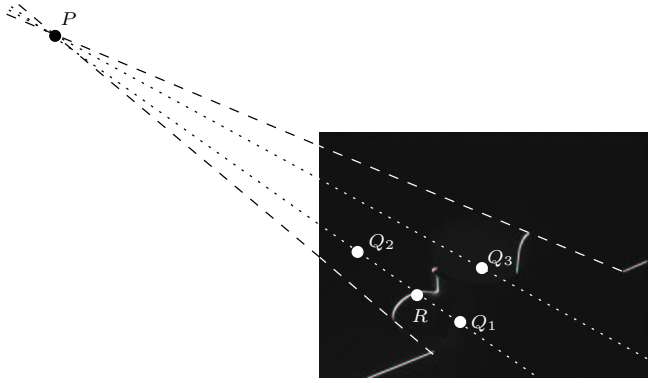


Figure 5.3: Finding pixels inside/outside the object

Our implementation creates binary images described above performing the following steps, assuming that, in addition to the input images containing projection of the laser light onto the object (Figure 5.4c), we also have taken an image of the projection of the laser light onto the acquisition space without any objects in it (Figure 5.4a) and that we know the world coordinates of the laser.

1. Image coordinates of the laser are calculated (point P in Figure 5.3).

2. The image of the acquisition space (Figure 5.4a) is binarized in the following way: starting from every pixel belonging to the laser line we "walk" toward the laser point P, painting all the encountered pixels white, resulting in image shown in Figure Figure 5.4b.

3. For every input image, an initial binary image is created, identical to the binarized image of the acquisition space (Figure 5.4b). In other words, we assume that the object lies entirely in the laser plane from Figure 5.4b.

4. Starting from every pixel belonging to the laser line in each input image we "walk" toward the laser point P, painting all the encountered pixels black. The resulting binary image is shown in Figure 5.4d.

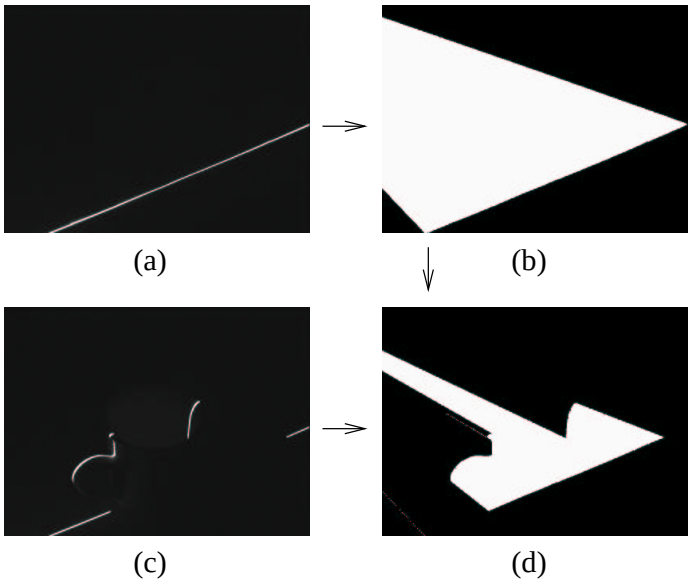


Figure 5.4: Binarization of Shape from Structured Light images

A line in an acquired image is typically 1–4 pixels wide, depending on the intensity of the laser light. Each point of the line (such as R in Figure 5.3) should represent the intersection of the corresponding laser ray and the surface of the object. Mathematically, each laser ray intersects the surface in at most one point (excluding special cases in which the object's surface at the point where the ray hits the surface is parallel to the ray). Therefore, as a preprocessing step, the laser line in each acquired image is thinned to 1-pixel width (Figure 5.5).

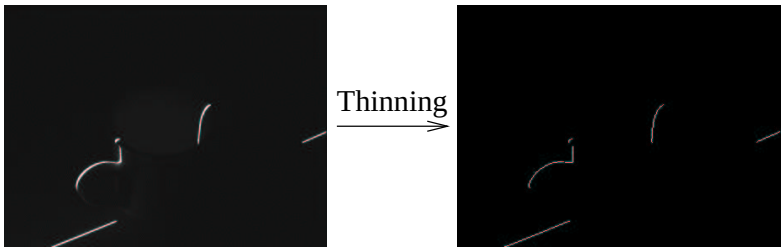


Figure 5.5: Laser line thinning

Note that in order to be able to calculate the image coordinates of the laser point P from Figure 5.3 we need to calibrate the camera and determine the world coordinates of the laser, as described in Sections 2.4 and 3.4, respectively.

5.3 Projection of an Octree Node into Image

In Section 3.2 we discussed the relevant coordinate systems, as shown in Figure 3.3. We still need to define where we place the octree. It is aligned with the world coordinate system. The center of the root node is identical to the origin of the world coordinate system. This is illustrated in Figure 5.6, which is identical to Figure 3.3 with the octree added. The initial size of the octree (i.e., the size of the root node) can be set manually

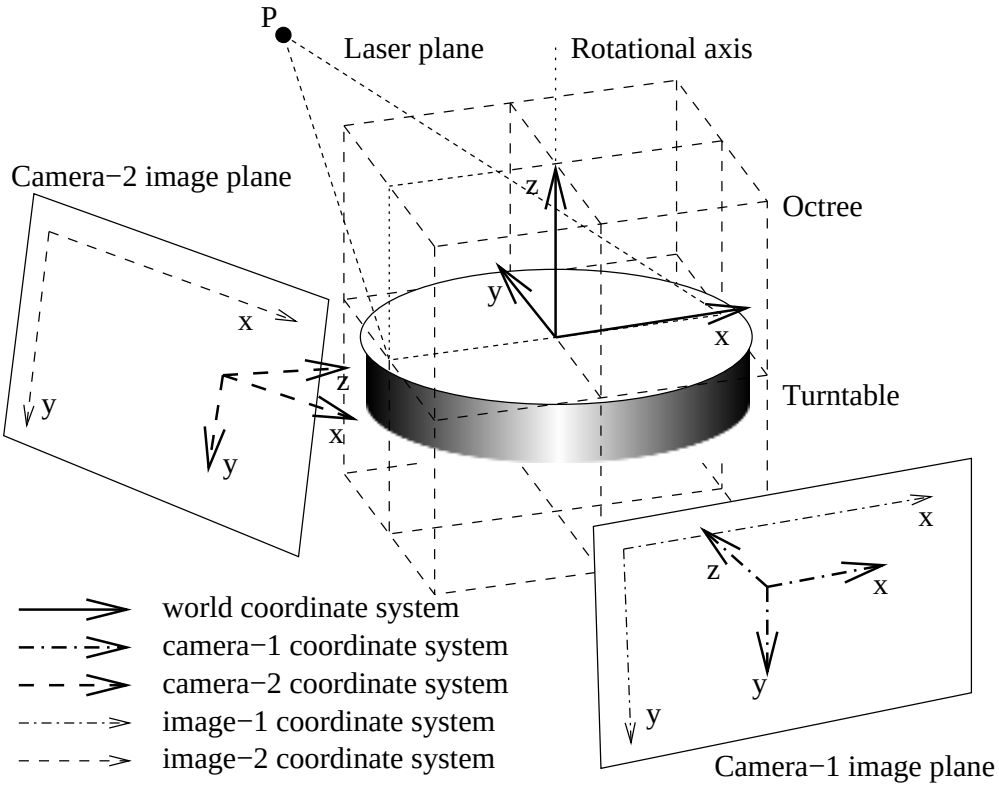


Figure 5.6: Placing the octree in world coordinate system

or calculated automatically by finding a cube rooted in the origin of the world coordinate system whose projection into all Shape from Silhouette images completely contains the silhouette of the object.

Knowing the world position and the size of the root node of the octree we can also calculate the world coordinates of the center or the vertices of any node at any level. The camera calibration described in Section 2.4 provides us with the parameters of the transformation between the world and the image coordinate system. If we denote the world coordinates of the point of the octree node we want to project into an image with (x_w, y_w), then, applying the for steps of world-to-image transformation described in Section 2.3, we can calculate its image coordinates (X_i, Y_i).

However, by acquiring images from multiple views by rotating the turntable around its rotational axis, our world coordinate system also rotates around its z axis in front

of the camera. Because of this, for every input image there is a new relation between the image and the initial world coordinate system. The camera calibration provides us with the relation between the rotated world coordinate system and the image coordinate system. If the rotation angle of the world coordinate system around its z axis was α (Figure 5.7) for the image we want to process, we have to take this angle α into account when performing world-to-image transformation.

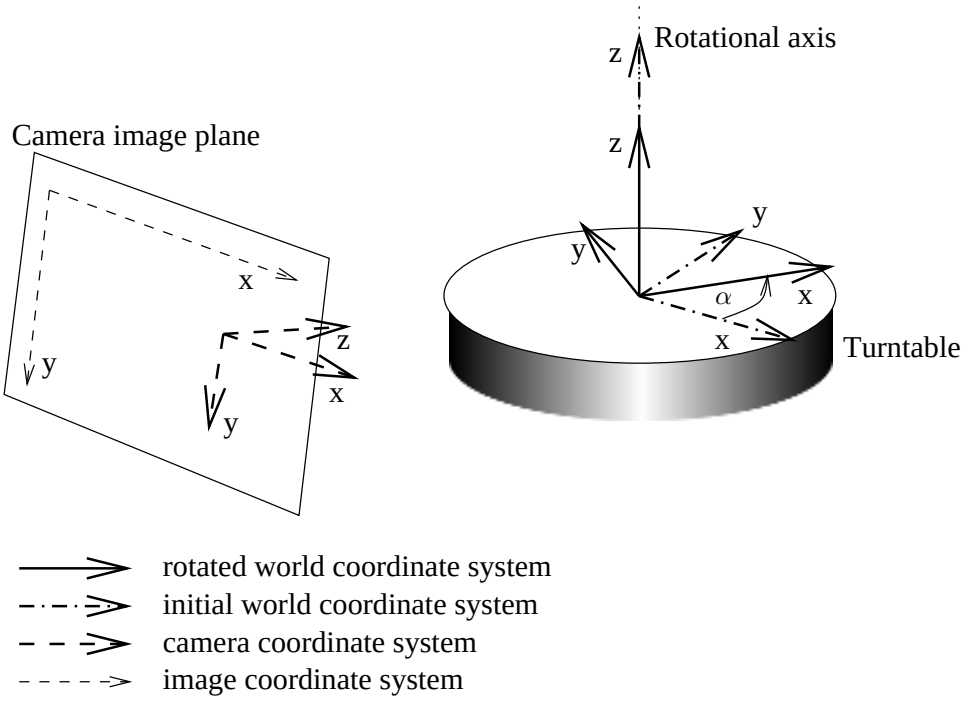


Figure 5.7: Initial and rotated world coordinate system

An input image for Shape from Silhouette contains three-dimensional information about the object (see conic volumes in Figure 4.7) while a Shape from Structured Light input image contains two-dimensional information only — the intersection of the object with the laser plane (see Figure 4.8a). Therefore, in the Shape from Silhouette part we calculate the α-rotated world coordinates of the node tested and then perform the world-to-image transformation described in Section 2.3. We do this for every input image (i.e., for every α) because we want to test whether the current node lies in *all* conic volumes. In contrast to this, in the Shape from Structured Light part we only want to test the node in the images representing the laser planes intersecting the node. There can be several planes intersecting the node and finding all of them can be a complex task. Instead, we decided to find one plane only, the one nearest to the center of the node and project the node into that plane before we perform the world-to-image transformation described in Section 2.3.

The remainder of this section describes the projection of a node into Shape from

Silhouette and Shape from Structured Light images in more detail, as well as finding the nearest laser plane.

5.3.1 Projection of a Node into Shape from Silhouette Image

For an input image for Shape from Silhouette taken from an angle α we have to rotate the initial world coordinate system in order to be able to apply the parameters obtained through camera calibration. If we apply Equation 2.12 from Section 2.3, the rotation between two coordinate systems with the angle α around the z axis is described through the rotation matrix R_Z (here given in homogeneous form):

$$R_Z = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

Through the camera calibration we obtained among others the parameters r_1–r_9 and T_x–T_z from Equation 2.14 defining the transformation between the (α-rotated) world and the camera coordinate system. For coordinates given in the initial world coordinate system we have to apply Equation 5.1 first, resulting in the modified world-to-camera transformation:

$$\begin{aligned} \begin{pmatrix} x_{w_{cam}} \\ y_{w_{cam}} \\ z_{w_{cam}} \\ 1 \end{pmatrix} &= \begin{pmatrix} r_1 & r_2 & r_3 & T_x \\ r_4 & r_5 & r_6 & T_y \\ r_7 & r_8 & r_9 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{w_{wrl}} \\ y_{w_{wrl}} \\ z_{w_{wrl}} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} r_1\cos\alpha - r_2\sin\alpha & r_1\sin\alpha + r_2\cos\alpha & r_3 & T_x \\ r_4\cos\alpha - r_5\sin\alpha & r_4\sin\alpha + r_5\cos\alpha & r_6 & T_y \\ r_7\cos\alpha - r_8\sin\alpha & r_7\sin\alpha + r_8\cos\alpha & r_9 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{w_{wrl}} \\ y_{w_{wrl}} \\ z_{w_{wrl}} \\ 1 \end{pmatrix} \end{aligned} \tag{5.2}$$

The other steps of the world-to-image transformation described in Section 2.3 are unaffected by the rotation angle α.

5.3.2 Finding the Nearest Laser Plane

When testing an octree node against Shape from Structured Light input images, only those images which represent laser planes that intersect the node are relevant for the test. The plane containing the node center is the most relevant one, because its intersection with the node is the largest among the planes intersecting the node. However, by having a limited number of input images, it is very likely that none of the corresponding laser planes contain the node center. Therefore, among all planes represented by the input images, we search for the one nearest to the node center. This is done by calculating the distance between the center of the node processed and all laser planes and selecting the plane for which this distance is minimal.

Section 2.5 explained how to find the distance D_Q between any point Q in 3D space and a given 2D plane, defined by its normalized orthogonal vector $\vec{n'}$ and the distance d to the origin (see Equation 2.25). According to our definition of relevant coordinate systems (see Figure 3.3), a Shape from Structured Light image taken with the initial world coordinate system represents the world x-z plane. An image taken with α-rotated world coordinate system (Figure 5.7) contains the world z-axis and the angle between this plane and the x-z plane is α. Applying Equation 2.25 for every laser plane acquired, we can calculate the distance of the center of the node processed and the laser plane. The laser plane for which this distance is minimal is the nearest plane to the node.

How do we find the vector $\vec{n'_\alpha}$ and the distance d_α of the plane taken from angle α? Let us denote this plane Π_α and take a look into Figure 5.8. Considering that all our laser

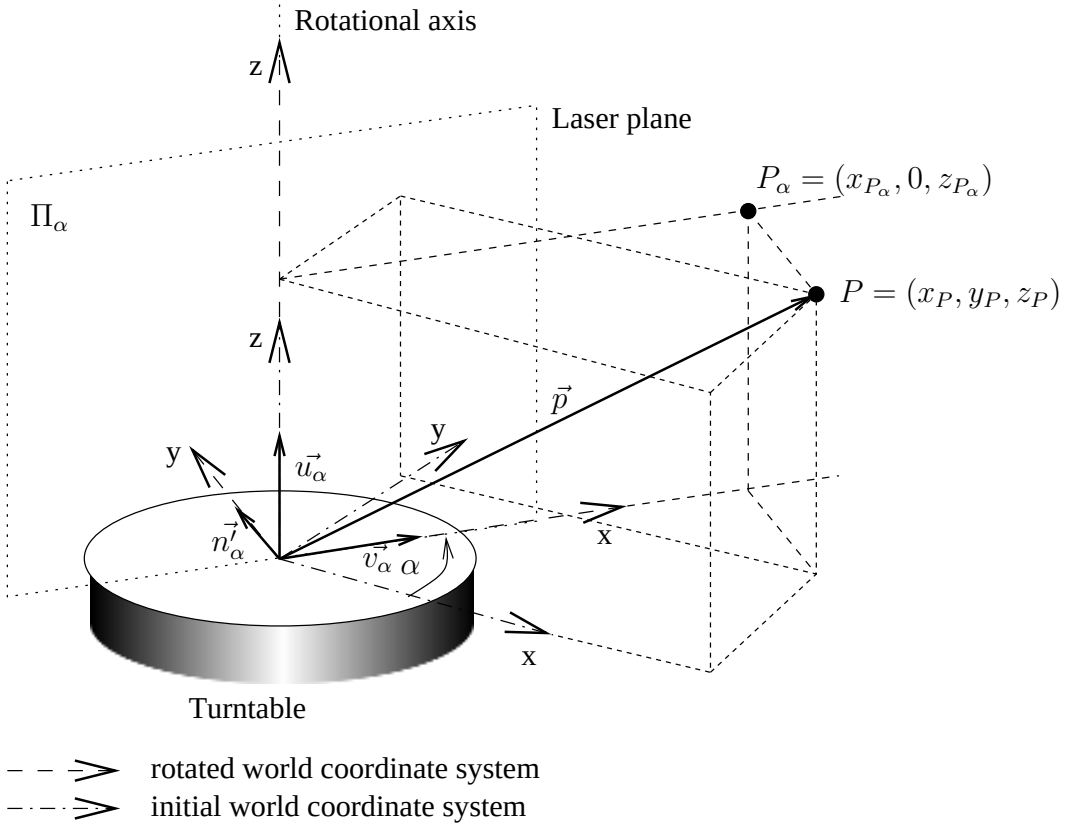


Figure 5.8: Parameterization of laser plane

planes contain the z axis of the world coordinate system, all of them contain its origin, which means $d = 0$ for all planes. In order to calculate $\vec{n'_\alpha}$ we need three non-collinear points on the plane Π_α (see Section 2.5). Finding two points is trivial — we can take any two points on the z axis, for example $(0, 0, 0)$ and $(0, 0, 1)$. For the third point we can set z to 0 and calculate x and y. Considering the angle α between Π_α and the world x-y

plane it is obvious that

$$z = 0 \Longrightarrow \begin{cases} x = r \cdot \cos\alpha & r \in \mathbf{R} \\ y = r \cdot \sin\alpha \end{cases} \tag{5.3}$$

We set r from Equation 5.3 to 1. Using these three points we now can build the vectors $\vec{u}$ and $\vec{v}$ from Equation 2.20 (now denoting them $\vec{u_\alpha}$ and $\vec{v_\alpha}$)

$$\vec{u_\alpha} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \vec{v_\alpha} = \begin{pmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{pmatrix} \tag{5.4}$$

and the vector $\vec{n_\alpha}$ using Equation 2.21

$$\vec{n_\alpha} = \vec{u_\alpha} \times \vec{v_\alpha} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{pmatrix} = \begin{pmatrix} \sin\alpha \\ -\cos\alpha \\ 0 \end{pmatrix} \tag{5.5}$$

Note that the length of $\vec{n_\alpha}$ is 1, i.e., it is already normalized, which means that the vector $\vec{n_\alpha'}$ we are looking for is

$$\vec{n_\alpha'} = \vec{n_\alpha} = \begin{pmatrix} \sin\alpha \\ -\cos\alpha \\ 0 \end{pmatrix} \tag{5.6}$$

Now we can calculate the distance D_α between any node with the center in point $C = (x_C, y_C, z_C)$ and the plane Π_α as

$$D_\alpha = \vec{c} \cdot \vec{n_\alpha'} = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} \cdot \begin{pmatrix} \sin\alpha \\ -\cos\alpha \\ 0 \end{pmatrix} = x_C \sin\alpha - y_C \cos\alpha \tag{5.7}$$

Among all planes Π_α the node processed is projected to the plane for which the absolute value of D_α is minimal.

5.3.3 Projection of a Node into Shape from Structured Light Image

Having found the node's nearest laser plane and its corresponding rotation angle α, we can proceed with the projection of the node into this plane. Note that the vectors $\vec{u_\alpha}$ and $\vec{v_\alpha}$ from Equation 5.4 and $\vec{n_\alpha'}$ from Equation 5.6 are the unit vectors of the z, x and y axes of the α-rotated world coordinate system (see Figure 5.8). The laser plane Π_α is the x-z plane of this α-rotated world coordinate system. This greatly simplifies the projection of a 3D point P (i.e., vector $\vec{p}$), given with its coordinates (x_P, y_P, z_P) in the initial world coordinate system, into the plane Π_α. If we denote the projected coordinates with $(x_{P_\alpha}, y_{P_\alpha}, z_{P_\alpha})$, then they can be calculated as follows:

$$\begin{pmatrix} x_{P_\alpha} \\ y_{P_\alpha} \\ z_{P_\alpha} \end{pmatrix} = \begin{pmatrix} \vec{p} \cdot \vec{v_\alpha} \\ 0 \\ \vec{p} \cdot \vec{u_\alpha} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{pmatrix} \\ 0 \\ \begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} x_P \cos\alpha + y_P \sin\alpha \\ 0 \\ z_P \end{pmatrix} \tag{5.8}$$

Now we can use the calibration parameters to perform the the four steps of world-to-image transformation as described in Section 2.3.

5.4 Intersection Test

The result of the projection of an octree node into the image plane are image coordinates of all of the vertices of the node's corresponding cube. In the general case, the projection of a node looks like a hexagon, as depicted in Figure 5.9(a). To find the hexagon corresponding to the eight projected vertices is a costly task, because it requires to determine which points are inside and which outside the hexagon, and there can be hundreds of thousands of octree nodes that need to be processed. It is much simpler (and therefore faster) to compare the bounding box of the eight points. Figure 5.9 shows a projected octree node and the corresponding bounding box. The bounding box is tested for intersection with

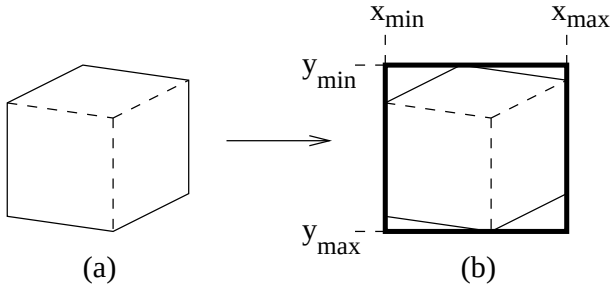


Figure 5.9: Projection of a node (a) and its bounding box (b)

the object's representation in the current input (binary) image. All image pixels within the bounding box are checked for their value, whether they have the value 0 (background) or 1 (object). The output of the intersection testing procedure is percentage of the pixels of the bounding box with value 1, i.e., percentage of pixels belonging to the object. If this percentage is equal or higher than a user definable threshold for black nodes, the node is marked as black, i.e., it belongs to the object (node 2 in Figure 5.10). If the percentage is smaller than or equal with a user definable threshold for white nodes, the node is marked as white, i.e., belonging to the background (node 1 in Figure 5.10). Otherwise, the node is marked as gray and it is divided into eight child nodes representing eight subcubes of finer resolution (node 3 in Figure 5.10). In the special case when the desired maximal octree depth has been reached and the intersection test would mark the node as gray, the node is simply marked as black instead.

One more issue needs to be mentioned. The calculated image coordinates of the cube's vertices can lie between two image pixels, and a pixel is the smallest testable unit for intersection testing. Which pixels are considered to be "within" the bounding box? Figure 5.11 illustrates our answer to this question. We decided to test only pixels that lie completely within the bounding box (Figure 5.11a), because that way the number of pixels that need to be tested is smaller than if we tested all pixels that are at least partly covered by the bounding box and it also makes sense to exclude the pixels at the border

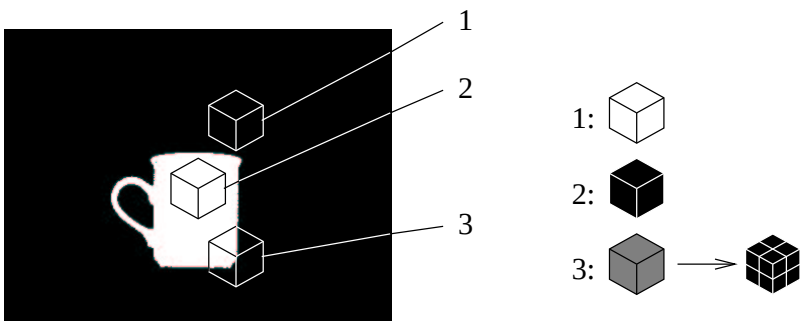
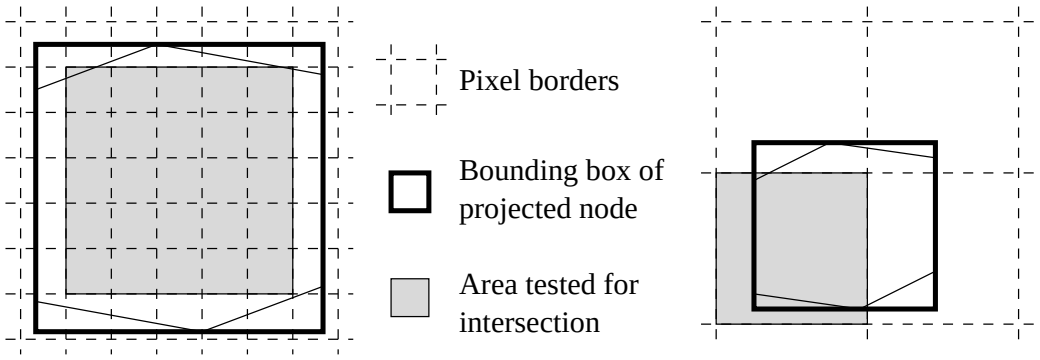


Figure 5.10: Intersection Test

of the bounding box, because most of them do not lie within the hexagon approximated by the bounding box. In the special case if there are no pixels that lie completely within the bounding box (Figure 5.11b) the pixel closest to the center of the bounding box is checked for the color.



(a) At least one pixel completely within bounding box

(b) No pixels completely within bounding box

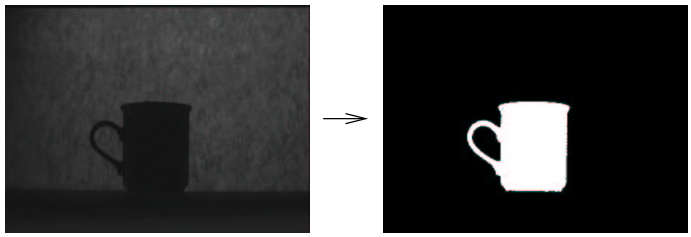
Figure 5.11: Selection of pixels for intersection test

5.5 Putting It All Together

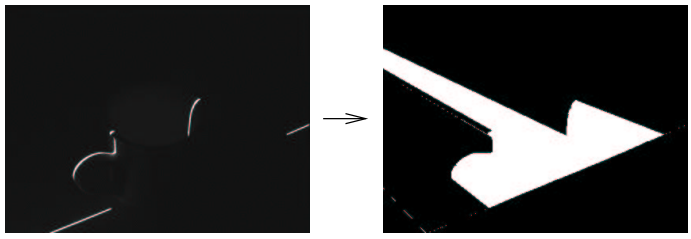
Our approach builds a 3D model of an object performing the following steps (illustrated in Figure 5.12):

1. Binarize the acquired images for both Shape from Silhouette and Shape from Structured Light as described in Section 5.2 (Figure 5.12a).

2. Build the initial octree, containing one single root node marked "black" (Figure 5.12b). This node is said to be at the level 0. Set the `current level` to 0.

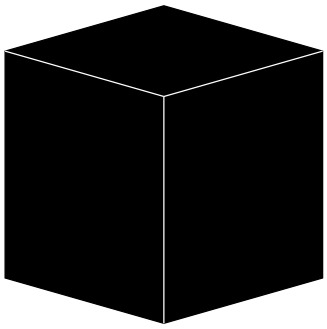


Shape from Silhouette

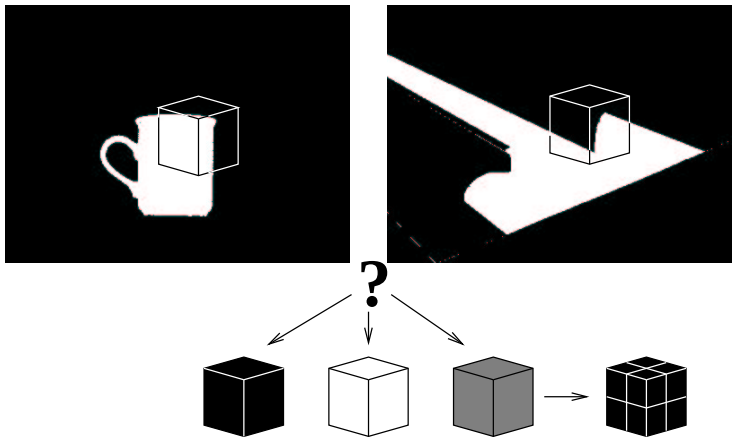


Shape from Structured Light

(a) Binarization of input images



(b) Initial octree



(c) Intersection testing

(d) Final model

Figure 5.12: Algorithm overview

3. All black nodes of the `current level` are assumed to be in a linked list. Set the `current node` to the first node in the list. If there are no nodes in the `current level`, the final model has been build so jump to Step 9. Otherwise, the `current node` needs to be projected into the relevant input images (Figure 5.12c), so continue with Step 4.

4. Find the two binarized Shape from Structured Light images representing the two laser planes nearest to the `current node` — one plane is the nearest among the images acquired with the turntable's rotation angle between 0° and 180° and the other the nearest among images taken with the angle between 180° and 360°. For

each of these intervals the nearest plane is found as described in Section 5.3.2. The separation into 2 intervals is done because if we use a single nearest plane, it could happen that the projection of the node lies completely in the occluded part of the image. Two nearest planes defined this way are almost identical, because they both contain the rotational axis of the turntable (because of the way we set the laser plane, see Figure 3.3) so if the nearest plane in the range $0° - -180°$ was with the angle α, then the nearest plane in the range $180° - -360°$ will be with the angle $\alpha + 180°$. This way we increase the chance that the node does not lie in the occluded area in at least one of the planes.

5. Project the `current node` into the two images found in Step 4, as described in Section 5.3.3, and for both images perform the intersection test described in Section 5.4. If at least one image says "this node is white", it is set to white. Otherwise, if at least one image says "this node is gray", it is set to gray and only if both images agree that the node is black, it stays black.

6. If the `current node` after Step 5 is not white, project it into all binarized Shape from Silhouette input images (as described in Section 5.3.1) and intersect it with the image silhouettes of the object (as described in Section 5.4). If at least one image says "this node is white", it is set to white. Otherwise, if at least one image says "this node is gray", it is set to gray and only if all images agree that the node is black, it stays black.

7. If the node is set to gray it is divided into 8 child nodes of the `current level` $+ 1$, all of which are marked "black"

8. Processing of the `current node` is finished. If there are more nodes in the `current level` set the `current node` to the next node and go back to Step 4. If all nodes of the `current level` have been processed, increment the `current level` and go to Step 3.

9. The final octree model has been built (Figure 5.12d).

By building a 3D model as described above the octree is traversed only once — each node, when being processed, is projected to all relevant input images and once it is done, it does not need any further processing. The octree is also processed in a level-by-level manner — all nodes of one level are completely processed before we go one with the next higher level. This is efficient because there are no unnecessarily processed nodes. To illustrate this, let us go back to the conic volumes in Figure 4.7. The volume in Figure 4.7c was built using two views only (from Figure 4.7a and 4.7b), but we can already see that the conic volumes in Figure 4.7a and 4.7b have many high level details which are completely cut by the intersection of the volumes. Therefore, it is better to consider all views while building each level so that the minimal possible number of high level nodes is processed. Intuitively described, by processing the octree in a level-by-level manner, we carve out a large block of space in the shape of the object modeled, removing larger pieces first and then going into finer and finer details, as illustrated in Figure 5.13. At Level 0 there is one node only, the root node, and it is gray. At Level 1 there are 8 nodes, all of them gray,

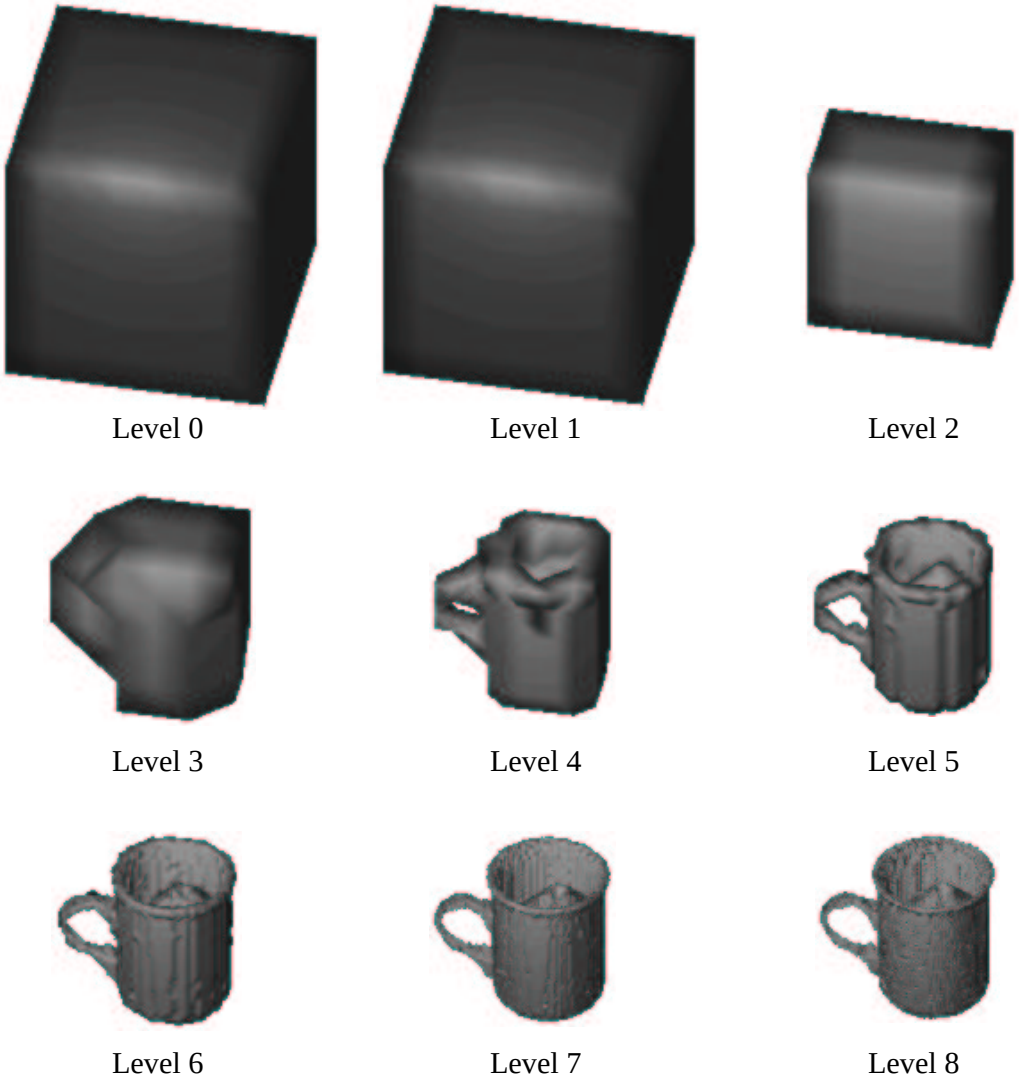


Figure 5.13: Level-by-level building of octree model

because each of the occupies partly the object and partly the background. Therefore, the models at Level 0 and Level 1 are identical. Starting from Level 2 some white nodes start showing up and from Level 3 the model starts shaping up in the form of the mug, resulting with a fine resolution model at Level 8.

5.6 Summary

This chapter presented our 3D modeling approach in detail — how the input images are acquired and processed and how an octree model of an object is built based on projection and intersection of its nodes with the object's representations in the images. For acquisition of multiple views of an object it has been mentioned that the turntable can be rotated by a constant angle between two views or that the angle can be calculated

automatically for each view. Automatic calculation of the next view is called Next View Planning and it is the topic of the following chapter.

Chapter 6

Next View Planning

This chapter covers the topic of Next View Planning (NVP) — it starts with introducing the problem of Next View Planning with a simple example in Section 6.1, followed by a general overview of the NVP problem in Section 6.2. Section 6.3 proposes a simple solution for our 3D modeling approach and our specific acquisition system, followed by the summary in Section 6.4.

6.1 Introduction

In order to create a complete three-dimensional model of an object based on its two-dimensional images, the images have to be acquired from different views. An increasing number of views generally improves the accuracy of the final 3D model but it also increases the time needed to build the model. The number of the possible views can theoretically be infinite. Therefore, it makes sense to try to reduce the number of views to a minimum while preserving a certain accuracy of the model, especially in applications for which the performance is an important issue.

One possibility for obtaining multiple views is to choose a fixed subset of possible views, usually with a constant step between two neighboring views, independent on the shape and the complexity of the object observed. This is illustrated in Figures 6.1a and 6.1b, which show a reconstruction of a corner of a square by drawing lines from the point O with a constant angle between two lines and connecting the points where the lines intersect the square. We can see that the corner reconstructed using 9 lines (Figure 6.1b) looks "better" that the one reconstructed using 5 lines (Figure 6.1a), but also that neither of these two methods was able to reconstruct the corner perfectly. In addition to this, some of the views (20° in Figure 6.1a and 10°, 20°, 30°, 60° and 70° in Figure 6.1b) could have been omitted — without them the reconstruction of the corner in Figures 6.1a and 6.1b would have been exactly the same.

This simple example illustrates the need for selection of views based on the features of the object — this is called *Next View Planning* (in short, *NVP*). For the square from Figure 6.1, if we had a way of selecting the significant views only, we could reconstruct

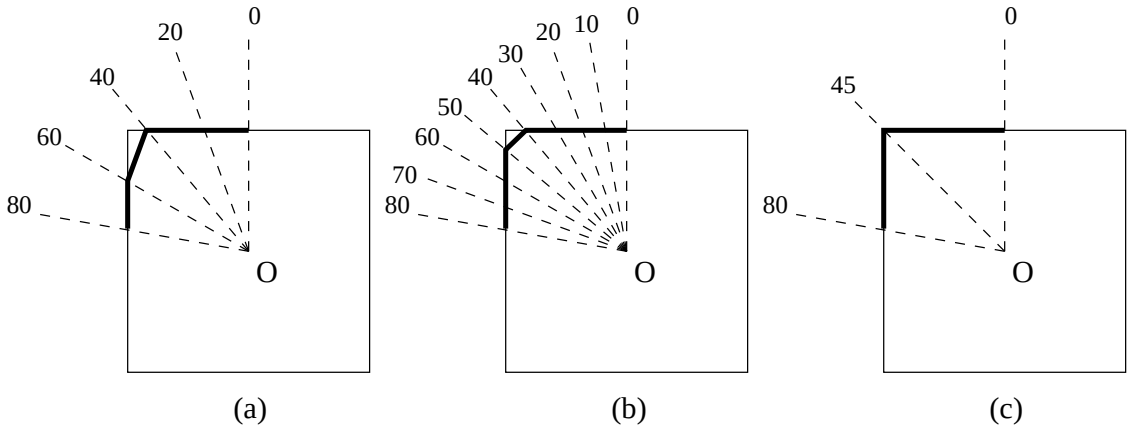


Figure 6.1: Reconstruction of a square corner

the corner of the square perfectly using 3 views only, as shown in Figure 6.1c.

6.2 Overview

The example from Figure 6.1 illustrates only one in a broad range of problems addressed by Next View Planning. A thorough survey of Next View Planning, also called *Sensor Planning*, is given in [TAT95]. The following sentence, taken from Tarabanis et al. [TAT95], summarizes the NVP problem: "Given the information about the environment (e.g., the object under observation, the available sensors) as well as the information about the task that the vision system is to accomplish (i.e., detection of certain object features, object recognition, scene reconstruction, object manipulation), develop strategies to automatically determine sensor parameter values that achieve this task with a certain degree of satisfaction". Following this definition, in order to design an NVP algorithm for a given computer vision task, one has to identify the sensor parameters which can be manipulated (e.g., the position of the camera) and define the "degree of satisfaction", i.e., construct a metrics for evaluation of the parameter values proposed. The number of parameters that can be manipulated is also called the number of *degrees-of-freedom*. Increasing number of degrees of freedom increases the complexity of an NVP algorithm.

There are several computer vision tasks which can incorporate an NVP problem, differing in the necessary amount of an a priori knowledge about the object, the sensors and the environment:

- *Object feature detection*: here the goal of NVP is to determine the sensor parameters values for which the particular features of a known object in an image satisfy certain constraints, such as being visible and in-focus [CK88, TTK91]. A considerable amount of a priori knowledge about the approximate pose of the object and the environment is required.

- *Visual inspection*: this is a sub-area of object feature detection — a typical task of visual inspection is to determine how accurately a particular object has been manufactured [TG95b, TG95a, MR95]. A nearly perfect estimate of the geometry and the pose of the object have to be known.

- *Model-based object recognition*: in this area NVP tries to find the sensor parameter values which make it possible to identify an object and/or estimate its pose in a most accurate and efficient way [KK94, KJV85]. Based on models of sensors and possible objects, a search on object's identity and pose is performed, usually using the hypothesize-and-verify method: in the first step, the hypotheses regarding the object's identity and pose are formed; then, these hypotheses are evaluated according to certain metrics; finally, the new sensor parameter values are proposed based on a given criterion until a stopping condition is met.

- *Scene or object reconstruction*: in this case, the task of NVP is to find the best values of the sensor parameters in order to build a model of an unknown scene or object [MC96, Con85, MB93, Pit99]. A model is built incrementally, guided by the information about the scene/object acquired to this point. Usually there is no a priori known scene information.

The approach presented in this work falls into the category of object reconstruction. For this task, many different NVP strategies have been developed. Here we give an overview of few.

Maver and Bajcsy [MB93] proposed an NVP algorithm for an acquisition system consisting of a light stripe range scanner and a turntable. They represent the unseen portions of the viewing volume as $2\frac{1}{2}$D polygons. The polygon boundaries are used to determine the visibility of unseen portions from all candidate next views. The view which can see the largest area unseen up to that point is selected as the next best view.

Connolly [Con85] used an octree to represent the viewing volume. An octree node close to the scanned surface was labeled as *seen*, a node between the sensor and this surface as *empty* and the remaining nodes as *unseen*. Next best view was chosen from a sphere surrounding the object. Connolly proposed two NVP algorithms: one called *planetarium*, which used a form of ray tracing to determine the number of unseen nodes from each candidate view and selected the one seeing the most unseen nodes, and a *normal* algorithm, which selected the next best view from 8 candidate positions only and did not take occlusions into account, and therefore was significantly faster.

Whaite and Ferrie [WF94] use the range data sensed so far to build a parametric approximate model of the object. The view from which the data fits the current model the worst is chosen as the next best view. This approach does not check for occlusions and does not work well with complex objects because of limitations of a parametric model.

Pito [Pit99] uses a range scanner which moves on a cylindrical path around the object. He partitions the viewing volume into its *seen* and *unseen* portions, and defines the sur-

face separating the two volume portions as *void surface*. This surface is approximated by a series of small rectangular oriented *void patches*. In his *positional space (PS)* algorithm, the next best view is chosen as the position of the scanner which samples as many void patches as possible while resampling at least a certain amount of the current model.

Liska [Lis99] uses a system consisting of two lasers projecting a plane onto the viewing volume and a turntable. The next best view (the next position of the turntable) is computed based on information from the current and the preceding scan. In each of the two scans the surface point farthest from the turntable's rotational axis is detected as well as the corresponding point in the other scan. The pair of points with the greater change in the distance from the rotational axis is used to determine whether the current turntable step should be enlarged or made smaller.

6.3 Our Approach

Since the Next View Planning was not the main focus of this thesis, the idea was to implement a simple and straight-forward NVP algorithm which will at least nearly preserve the accuracy of models built using all possible views while reducing the number of views significantly. In most of object reconstruction tasks which involve some kind of Next View Planning, the NVP algorithm is part of the model building process and it is guided by some features of the partial model built based on preceding views. In our 3D modeling approach the acquisition of multiple views of an object and the actual object reconstruction are separated tasks — the modeling algorithm takes the images acquired as input and does not perform any view planing itself (see Section 5.5). Therefore, our goal was to design an NVP algorithm which does not need the partial model but uses only the features of the images acquired.

As described in Chapter 3, our acquisition system consists of a turntable, two cameras and a laser (Figure 3.1). The cameras and the laser are fixed while the turntable can rotate around its rotational axis. That means, our system has *one* degree of freedom. As noted in Section 3.1, the minimal rotation angle of the turntable is 1°. Therefore, the maximal number of views for our system is 360. With one degree of freedom and 360 possible views our acquisition system is fairly simple from the NVP point of view. Having the additional constraint of using the features of the images only, we propose a simple approach which takes only the current and the preceding image to decide what the next rotational step of the turntable will be. It defines a normalized metrics for comparison of the current and the preceding image. If the change is less than or equal to the maximal allowed change then the step is doubled. If the change is higher than the maximal change, then the current image is discarded and the turntable moves back by half the current step. In special cases where doubling the step exceeds the maximum or halving the step falls below the minimum, the new step is set to the maximum or minimum, respectively. The remainder of this section describes the metrics for comparison of images in detail and gives a step-by-step description of the NVP algorithm proposed.

6.3.1 Metrics for Comparison of Shape from Silhouette Images

The only information provided by a pixel in a silhouette image (see Figure 5.1a) is whether the pixel represents the object or the background. Following the notation common in NVP, we define a pixel representing the object as *seen* and a pixel representing the background as *empty*. Note that in a silhouette image there are no occlusions — the value of a pixel depends only on whether in the conic volume defined by the pixel there is a 3D point belonging to the object. Therefore, there can not be any *unseen* pixels, i.e., pixels for which we can not be sure whether they should be marked as seen or empty. In a binarized silhouette image (see Figure 5.2d) all white pixels are seen and all black pixels empty. Therefore, our NVP algorithm binarizes an acquired image in the same way as described in Section 5.2.1 and compares two binary images in the following way (illustrated in Figure 6.2): it counts all pixels which are seen in one and empty in the other image; in order to normalize this value, it is divided by the number of pixels which are seen in at least one of the images. With this metrics definition, if two silhouette images are identical, the

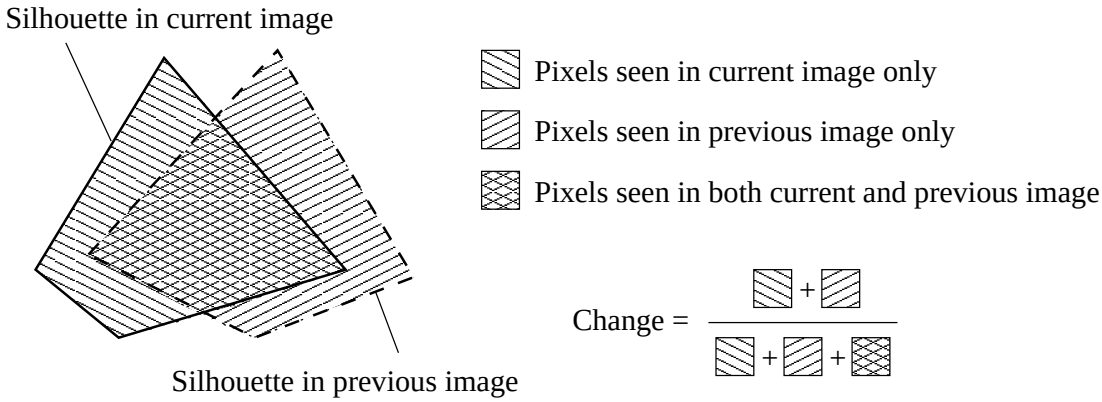


Figure 6.2: Change between two silhouette images

change is 0, and if the silhouettes do not intersect at all, it is 1.

Note that calculating the change used features of the images only and none of the information about the geometry of the acquisition system. This means that the system does not need to be calibrated prior to applying the NVP algorithm.

6.3.2 Metrics for Comparison of Shape from Structured Light Images

For Shape from Structured Light images we follow the same idea — we mark the pixels of the current and the preceding image as *seen*, *empty* or *unseen*, and count pixels which are seen in one and empty in the other image. A Shape from Structured Light input image contains a curve representing the intersection of the laser plane and the object (see Figure 5.1b). How do we decide which pixels are seen/empty/unseen? If we denote the source point of the laser with P and image pixels with Q_i (Figure 6.3), and draw a line

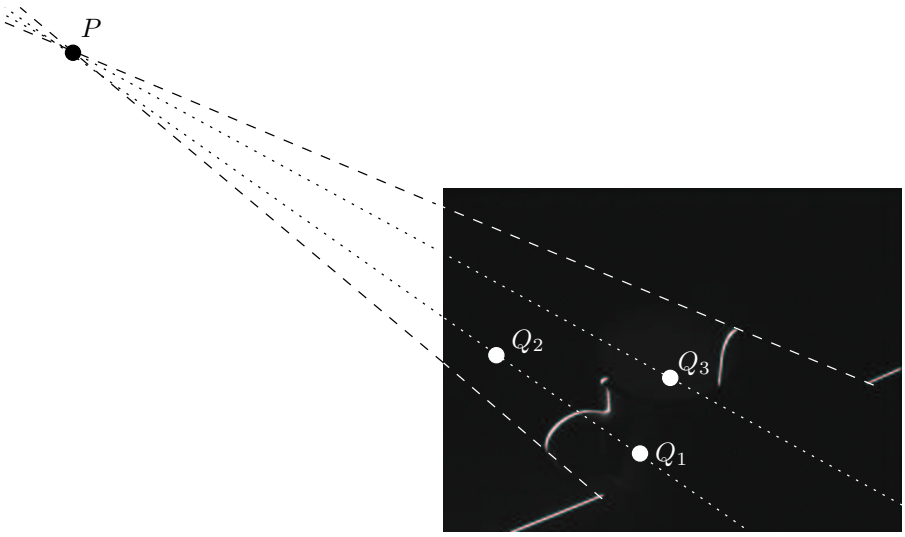


Figure 6.3: Seen, empty and unseen pixels in laser images

from P going through Q_i, we can differentiate between three types of points: if this line intersects the laser curve before it reaches Q_i (or exactly at Q_i), then Q_i is *below* the surface of the object and we mark it as seen (point Q_1 in Figure 6.3); if the line intersects the laser curve after the point Q_i, then Q_i is *above* the object's surface and we mark it as empty (Q_2 in Figure 6.3); finally, if the line does not intersect the laser curve at all, then Q_i is occluded by a part of the object outside the current laser plane and we can not say whether it is above or below the surface, so we mark it as unseen (Q_3 in Figure 6.3).

In order to perform the marking of pixels, our NVP algorithm processes the images acquired in a way very similar to binarization of laser images described in Section 5.2.2:

1. Image coordinates of the laser are calculated (point P in Figure 6.3).

2. The image of the acquisition space (Figure 6.4a) is binarized in the following way: starting from every pixel belonging to the laser line we "walk" toward the laser point P, painting all the encountered pixels *gray* (unseen), resulting in image shown in Figure 6.4b.

3. For the image acquired, an initial marking image is created, identical to the resulting image of the acquisition space from the previous step (Figure 6.4b).

4. Starting from every pixel belonging to the laser line in the acquired image we "walk" *toward* the laser point P, painting all the encountered pixels *black* (empty), and *away* from the point P, painting all the encountered pixels *white* (seen). The resulting image is shown in Figure 6.4d.

Having processed the current and the preceding image this way, our NVP algorithm compares these two images by counting pixels which are seen in one and empty in the other image. This number is normalized by dividing it with the number of pixels which are seen

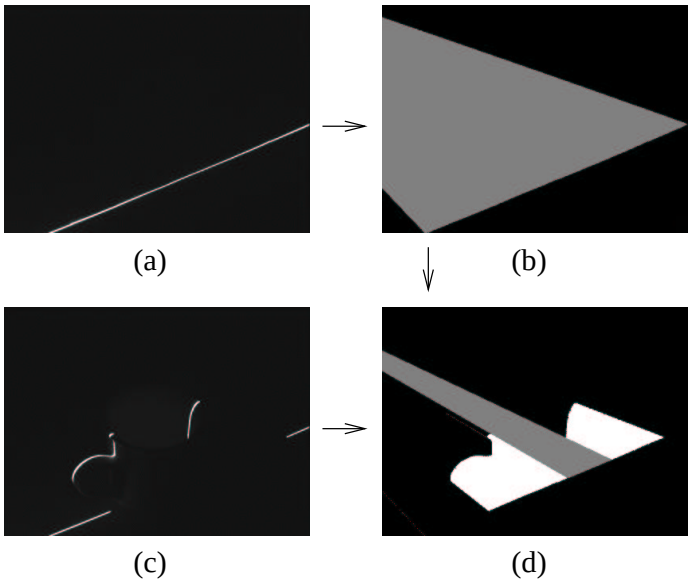


Figure 6.4: Processing a Shape from Structured Light image for NVP

in at least one of the two images, but not unseen in the other. In other words, because of uncertainty associated with the unseen pixels, they are completely disregarded by our NVP algorithm.

In order to mark the image pixels as described above, the NVP algorithm needs to know the laser position as well as how to transform the image to the world coordinates. Therefore, for Shape from Structured Light the acquisition system needs to be calibrated prior to applying the NVP algorithm.

6.3.3 Step-by-Step Description

To describe our NVP approach in detail, we use the following parameter notation:

- Angle steps: α_{min}, α_{max}, α_{init}, α_{curr} — the minimal, maximal, initial and the current step.
- Images acquired: I_n denotes the n^{th} image taken.
- The change between two images: C_{max}, C_{curr} — the maximal allowed and the current change.

For both Shape from Silhouette and Shape from Structured Light images, our NVP approach performs these steps:

1. Parameters are initialized. The users sets the initial step α_{init} and the maximal step α_{max} ($\alpha_{init} \leq \alpha_{max}$), as well as the maximal allowed change C_{max} between two subsequent images. This change is assumed to be normalized, i.e., $0 \leq C_{max} \leq 1$. The minimal step α_{min} is implied by the resolution of the turntable (1° for our turntable).

2. The first image I_1 is taken. The current step α_{curr} is set to the initial value: $\alpha_{curr} = \alpha_{init}$. Number of acquired views n is set to one: $n = 1$.

3. If the turntable already has made a complete revolution of 360°, we are done. Otherwise, the turntable is rotated by the angle α_{curr}, the image I_{n+1} is taken and we continue with Step 4.

4. The change C_{curr} between the images I_{n+1} and I_n is evaluated. If $C_{curr} \leq C_{max}$ or $\alpha_{curr} = \alpha_{min}$ the image I_{n+1} is accepted, jump to Step 6. Otherwise the image I_{n+1} is discarded, continue with Step 5.

5. The step α_{curr} is halved: $\alpha_{curr} = \frac{1}{2} \cdot \alpha_{curr}$. If α_{curr} became smaller than α_{min} it is set to α_{min}. The turntable is rotated by $-\alpha_{curr}$ (i.e., back by the half of the previous step). Go back to Step 4.

6. Increment the image counter n by one and double the step α_{curr}: $n = n + 1$, $\alpha_{curr} = 2 \cdot \alpha_{curr}$. Jump back to Step 3.

6.4 Summary

This chapter introduced and discussed the problem of Next View Planning in general and presented a simple approach for a turntable-based acquisition system with one degree of freedom. It concludes the part of this thesis describing the theoretical background and the algorithms developed. The following chapter presents experimental results.

Chapter 7

Results

This chapter presents the tests performed on the 3D modeling method described in this work and analyzes their results. Sections 7.1 and 7.2 analyze camera and laser calibration errors, respectively. Tests and results with synthetic data are described in Section 7.3, with real data in Section 7.4 and experiments with Next View Planning in Section 7.5. Section 7.6 discusses the performance issues, followed by the summary in Section 7.7.

7.1 Analysis of Camera Calibration Errors

Calibration of both cameras was performed with a data set of 75 points whose world coordinates (x_w, y_w, z_w) are known and the corresponding image coordinates (X_i, Y_i) are measured. Once the calibration is completed and the model of the camera has been built, we can estimate its accuracy by using the camera model to perform world-to-image or image-to-world projection of the points from the calibration data set and measuring the distance between the projection of the points and their ideal coordinates.

For both cameras we measure this distance for each point from the calibration data set and calculate the mean error, the maximal error and the standard deviation of the error. Table 7.1 summarizes the results for the Shape from Silhouette camera and Table 7.2 for the Shape from Structured Light camera. For the *distorted image plane error* the ideal image coordinates of calibration points are compared with the image coordinates obtained through applying the camera model on the calibration points' world coordinates, taking lens distortion into account. *Undistorted image plane error* is the same, with the difference that lens distortion is ignored. For the *object space error* the camera model is applied to the ideal image coordinates of calibration points and the calculated world coordinates are compared with the ideal ones.

The 3D modeling method presented in this work performs world-to-image transformations of the vertices of the octree nodes. Therefore, the distorted or undistorted image plane error, depending on whether lens distortion is taken into account or not, is the relevant error measure for our approach. For both cameras the average error was 0.5 pixel or less, which is sufficient for our approach, because the smallest unit processed in an image is 1 pixel.

type of error	*mean*	*max*	*standard deviation*
distorted image plane	0.286592 pix	0.641272 pix	0.125903 pix
undistorted image plane	0.287583 pix	0.642413 pix	0.126503 pix
object space	0.129321 mm	0.265288 mm	0.055917 mm

Table 7.1: Camera calibration errors — Shape from Silhouette

type of error	*mean*	*max*	*standard deviation*
distorted image plane	0.496943 pix	1.203704 pix	0.256143 pix
undistorted image plane	0.502029 pix	1.224840 pix	0.258809 pix
object space	0.153209 mm	0.334654 mm	0.073592 mm

Table 7.2: Camera calibration errors — Shape from Structured Light

7.2 Analysis of Laser Calibration Errors

The position of the laser is calculated by intersecting rays coming from the laser (see Section 3.4). A ray can be defined through two points belonging to it. For error estimation we measured four rays (i.e., eight points) and intersected each two, thus obtaining six results as the position of the laser. The mean error, the maximal error and the standard deviation of the error are summarized in Table 7.3, taking the mean position as the exact one. The mean position is about 373 mm away from the origin of the world coordinate system where the objects are placed. As Table 7.3 indicates, the error of the estimated

mean error	*max error*	*standard deviation*
0.956311 mm	2.492263 mm	0.881400 mm

Table 7.3: Laser calibration errors

position of the laser is about 1 mm. In our experiments the objects are placed around the origin of the world coordinate system and for all objects their surface was at least 300 mm away from the laser. The position of the laser is used for the binarization of Shape from Structured Light images (see Section 5.2.2) in order to find the pixels inside and outside the object (see Figure 5.3). At the distance of 300 mm or more, an error of 1 mm of the position of the laser is negligible.

7.3 Synthetic Objects

For tests with synthetic objects we can build a model of a virtual camera and laser and create input images such that the images fit perfectly into the camera model. This way we can analyze the accuracy of the constructed models without impact of camera calibration errors. The parameters and the position of the camera and the laser are arbitrary, so we choose realistic values. We assume having a virtual camera with focal length $f = 20\ mm$,

placed on the y axis of the world coordinate system, 2000 mm away from its origin (Figure 7.1). We set the distance between two sensor elements of the camera to $d_x = d_y = 0.01$ mm. The laser is located on the z axis of the world coordinate system, 850 mm away from its origin, and the turntable 250 mm below the x-y plane of the world coordinate system, with its rotational axis identical to the z world axis, as shown in Figure 7.1. We

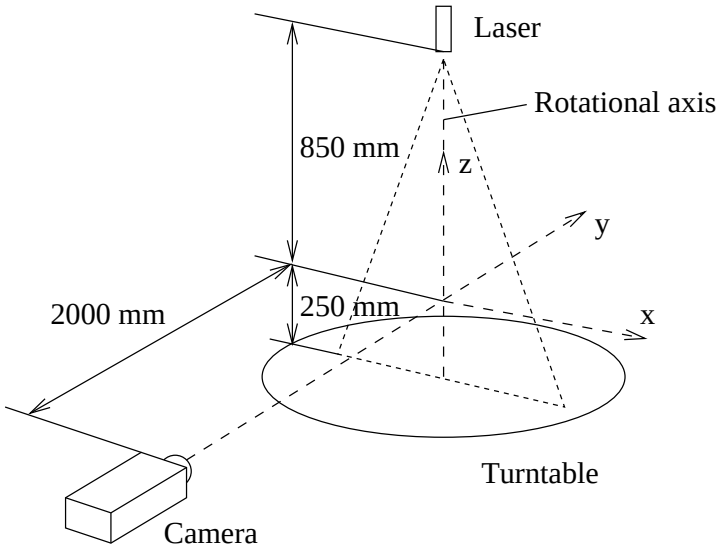


Figure 7.1: Virtual acquisition system

build input images with size 640×480 pixels, in which 1 pixel corresponds to 1 mm in the x-z plane of the world coordinate system.

Having built the camera model and the input images we can test our 3D modeling algorithms with varying modeling parameters. As the measure of the accuracy of the models we compare the size (width, height and length) and the volume of the model with the size and the analytical volume of the object.

7.3.1 Synthetic Sphere

As the first synthetic object we create a sphere with radius $r = 200$ mm, shown in Figure 7.2a. If we place the center of the sphere in the origin of the world coordinate system (see Figure 7.1), the sphere will look the same from all possible views. For our virtual acquisition system we can assume having neither camera nor light occlusions and we can construct perfect input images of the sphere (Figure 7.2b and c) which can be used for any view. Note that the image from Figure 7.2c can not be obtained using the laser from Figure 7.1. Instead, we assume seeing the complete profile of the sphere, in order to be able to reconstruct the complete object using Shape from Structured Light only. Since the sphere does not contain any cavities, Shape from Silhouette can also reconstruct it completely. Therefore, we can measure the accuracy of each of the methods independently,

as well as of the combined method.

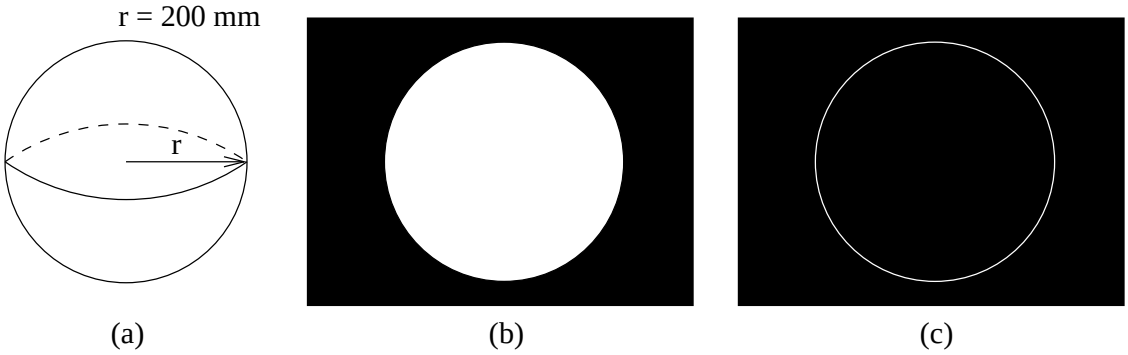


Figure 7.2: Synthetic sphere (a) and an input image for Shape from Silhouette (b) and Shape from Structured Light (c)

In the first test we build models using 360 views with the constant angle of 1° between two views, while increasing octree resolution. Table 7.4 summarizes the results, giving the size, the computed volume and the relative error of volume computation for each model built. Figure 7.3 visualizes these models. The models built with octree resolution of 2^3, 4^3, 8^3 and 16^3 are shown only once in Figure 7.3, because the models produced by the two methods were identical.

octree	*voxel size*	*method*	*dimensions (mm)*	*volume (mm^3)*	*error*
—	—	analytic	$400 \times 400 \times 400$	33 510 322	—
2^3	256 mm	Silhouette	$512 \times 512 \times 512$	134 217 728	+300.52%
		Structured Light	$512 \times 512 \times 512$	134 217 728	+300.52%
4^3	128 mm	Silhouette	$512 \times 512 \times 512$	117 440 512	+250.46%
		Structured Light	$512 \times 512 \times 512$	117 440 512	+250.46%
8^3	64 mm	Silhouette	$512 \times 512 \times 512$	60 817 408	+81.49%
		Structured Light	$512 \times 512 \times 512$	60 817 408	+81.49%
16^3	32 mm	Silhouette	$448 \times 448 \times 448$	46 661 632	+39.25%
		Structured Light	$448 \times 448 \times 448$	46 661 632	+39.25%
32^3	16 mm	Silhouette	$416 \times 416 \times 416$	38 666 240	+15.39%
		Structured Light	$416 \times 416 \times 416$	39 321 600	+17.34%
64^3	8 mm	Silhouette	$400 \times 400 \times 400$	35 241 984	+5.17%
		Structured Light	$400 \times 400 \times 400$	35 868 672	+7.04%
128^3	4 mm	Silhouette	$400 \times 400 \times 400$	33 786 880	+0.83%
		Structured Light	$400 \times 400 \times 400$	34 352 640	+2.51%
256^3	2 mm	Silhouette	$396 \times 396 \times 400$	33 034 528	-1.42%
		Structured Light	$400 \times 400 \times 400$	33 605 888	+0.29%
512^3	1 mm	Silhouette	$396 \times 396 \times 400$	32 936 452	-1.71%
		Structured Light	$400 \times 400 \times 400$	33 605 248	+0.28%

Table 7.4: Reconstruction of synthetic sphere with increasing octree resolution

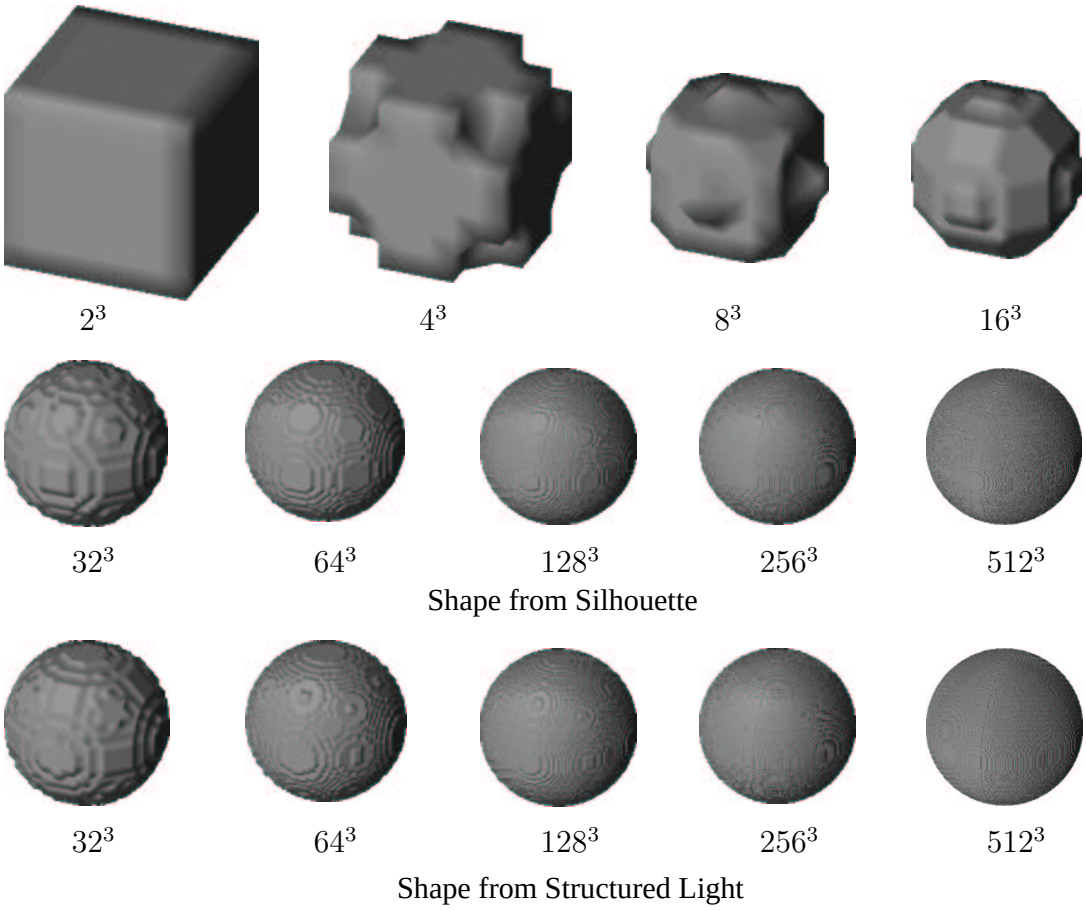


Figure 7.3: 3D models of synthetic sphere with increasing octree resolution

In the second test we build models with an increasing number of views and a constant octree resolution. The idea is to measure the impact of the number of views on the accuracy of the models. Since the number of possibly useful views increases with an increasing octree resolution, this resolution should be as high as possible. For our virtual acquisition system from Figure 7.1 and the synthetic input images from Figure 7.2 the maximal reachable octree resolution is 512^3, because the projection of higher resolution octree nodes into the image plane would be less than 0.5×0.5 pixels, and 1 square pixel is the smallest testable area in our modeling algorithm. However, considering that the models with the resolution 256^3 were insignificantly different from models with the resolution 512^3, but two times faster to build, in this test we use the octree resolution of 256^3. The angle between two neighboring views is always constant. Voxel size is 2 *mm* for all models. The results are summarized in Table 7.5 and the models shown in Figure 7.4. The column *error 1* in Table 7.5 refers to the error of the computed volume relative to the analytic volume and the column *error 2* to the error relative to the model built with 360 views and the same octree resolution of 256^3.

For the synthetic sphere, the models built using the combined method were identical

views	*method*	*dimensions (mm)*	*volume (mm^3)*	*error 1*	*error 2*
—	analytic	$400 \times 400 \times 400$	33 510 322	—	—
360	Silhouette	$396 \times 396 \times 400$	33 034 528	-1.42%	—
	Structured Light	$400 \times 400 \times 400$	33 605 888	+0.29%	—
4	Silhouette	$400 \times 400 \times 400$	38 218 688	+14.05%	+15.69%
	Structured Light	$404 \times 404 \times 400$	42 650 944	+27.28%	+26.92%
10	Silhouette	$400 \times 408 \times 400$	33 494 656	-0.05%	+1.39%
	Structured Light	$400 \times 420 \times 400$	34 749 408	+3.70%	+3.40%
20	Silhouette	$400 \times 400 \times 400$	33 230 464	-0.83%	+0.59%
	Structured Light	$400 \times 400 \times 400$	33 883 968	+1.12%	+0.82%
30	Silhouette	$400 \times 396 \times 400$	33 183 792	-0.97%	+0.45%
	Structured Light	$400 \times 400 \times 400$	33 726 304	+0.64%	+0.36%
60	Silhouette	$396 \times 396 \times 400$	33 097 536	-1.23%	+0.19%
	Structured Light	$400 \times 400 \times 400$	33 634 944	+0.37%	+0.09%
90	Silhouette	$396 \times 396 \times 400$	33 067 552	-1.32%	+0.10%
	Structured Light	$400 \times 400 \times 400$	33 617 440	+0.32%	+0.03%
180	Silhouette	$396 \times 396 \times 400$	33 048 448	-1.38%	+0.04%
	Structured Light	$400 \times 400 \times 400$	33 608 320	+0.29%	+0.01%

Table 7.5: Reconstruction of synthetic sphere with increasing number of views

to the models obtained using Shape from Silhouette only. Therefore, these models and their statistics are not shown in Figures 7.3 and 7.4 and Tables 7.4 and 7.5.

7.3.2 Synthetic Cone

As the second synthetic object we create a cone, having a cavity in the shape of a smaller cone (Figure 7.5a). We place the cone's axis of rotational symmetry in the rotational axis of the turntable and its base in the turntable's x-y plane (see Figure 7.1). With the virtual camera and laser placed as shown in Figure 7.1 we can construct input images fitting perfectly into the camera model. An input image for Shape from Silhouette is shown in Figure 7.5b and an input image for Shape from Structured Light in Figure 7.5c.

In this case none of the two algorithms can reconstruct the complete object — Shape from Silhouette can not see the cavity inside the cone and Shape from Structured Light can not see the outer cone shape because of light occlusions. Figure 7.6 shows the models constructed by using one of the underlying methods only.

Again, in the first test we build models of the cone using 360 views, while increasing octree resolution. In addition to measuring the accuracy of the computed volume of the complete object, we also measure the accuracy of the computed volume of the outer cone (reconstructed by Shape from Silhouette) and of the inner cone (carved out by Shape from Structured Light). Table 7.6 summarizes the results and Figure 7.7 shows the models built.

In the second test all models are built with the same octree resolution of 256^3 and voxel size of 2 mm, while increasing the number of views. The accuracy of the computed volume is measured for the outer and the inner cone, and for the complete object. The

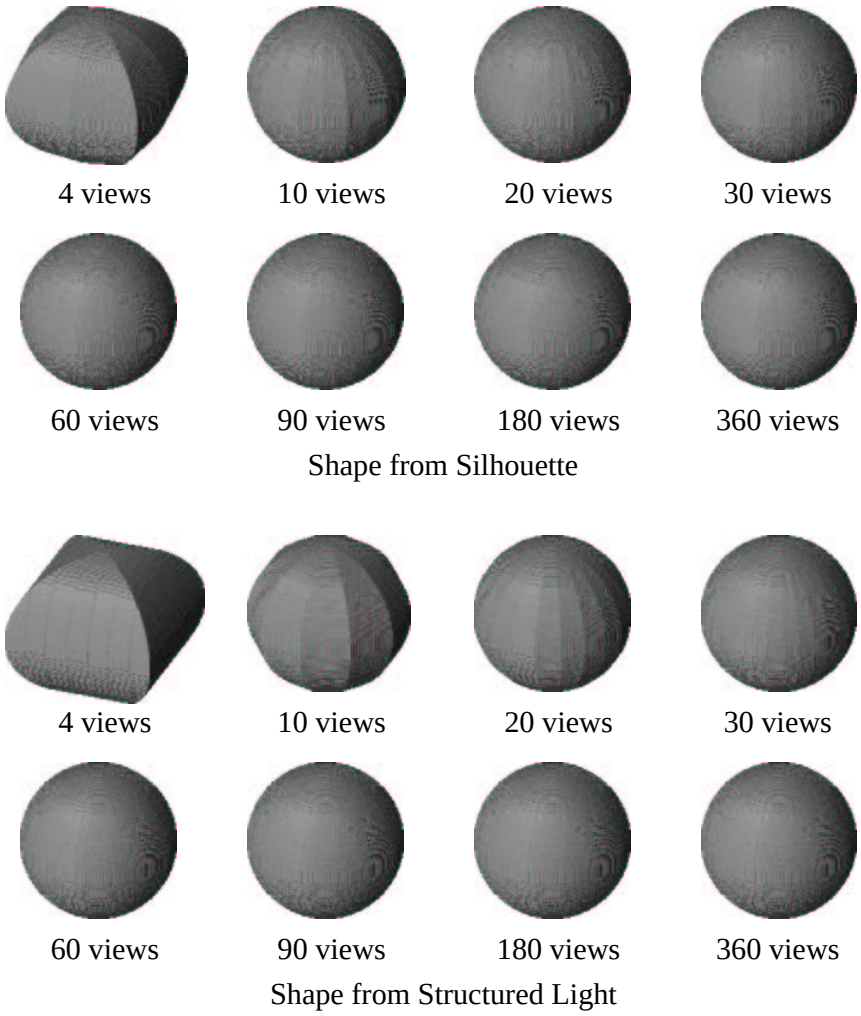


Figure 7.4: 3D models of synthetic sphere with increasing number of views

results are summarized in Table 7.7 and the constructed models shown in Figure 7.8. Figure 7.4. The column *error 1* in Table 7.5 refers to the error of the computed volume relative to the analytic volume and the column *error 2* to the error relative to the model built with 360 views and the same octree resolution of 256^3.

7.3.3 Synthetic Cuboid

As the third and final synthetic object we create a cuboid with the size $100 \times 70 \times 60\ mm$, which is equal to the size of the real cuboid we used in test with real objects, described later in this section. The camera and the laser from our virtual acquisition system from were placed closer to the object than shown in Figure 7.1, because otherwise the cuboid would appear small in the input images. The cuboid and a sample Shape from Silhouette and Shape from Structured Light image are shown in Figure 7.9.

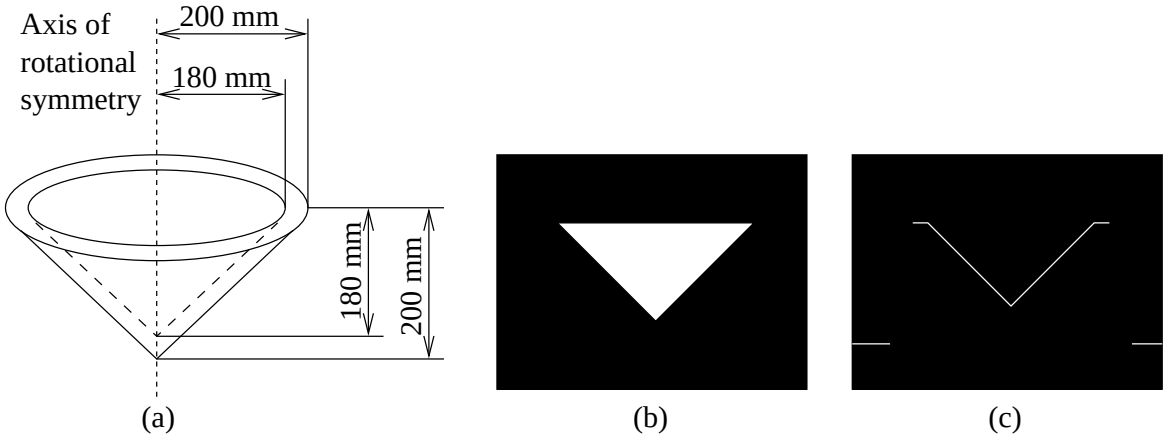


Figure 7.5: Synthetic cone (a) and an input image for Shape from Silhouette (b) and Shape from Structured Light (c)

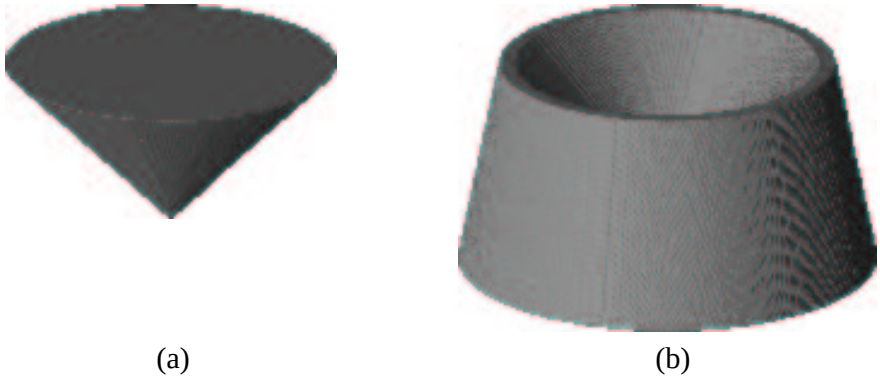


Figure 7.6: Models of the cone using either Shape from Silhouette (a) or Shape from Structured Light only (b)

Also for this object neither of the two methods alone was able to reconstruct the object completely, as illustrated in Figure 7.10 — Shape from Silhouette could not recover the upper flat surface of the cuboid (Figure 7.10a) and Shape from Structured Light could not see the four sides (Figure 7.10b) because of light occlusions.

In the first test of the combined method, we build models of the cuboid using 360 silhouette and 360 laser light views. Table 7.8 summarizes the results and Figure 7.11 shows the models built. The maximal octree resolution reached was 128^3, because at that depth there were no more gray nodes.

By using 360 views for both Shape from Silhouette and Shape from Structured Light, a perfect model of the cuboid was built, without any dimension or volume errors. In the second test, we vary the number of views while setting the maximal octree resolution to 256^3 and voxel size to 0.5 mm and compare the models built with the analytical object.

octree	*voxel size*	*method*	*volume* (mm^3)	*error*
—	—	analytic (outer cone)	8 337 580	—
		analytic (inner cone)	6 107 256	—
		analytic (complete object)	**2 270 324**	—
2^3	256 *mm*	Silhouette (outer cone)	67 108 864	+704.90%
		Structured Light (inner cone)	0	-100.00%
		combined (complete object)	**67 108 864**	**+2 855.92%**
4^3	128 *mm*	Silhouette (outer cone)	41 943 040	+403.60%
		Structured Light (inner cone)	0	-100.00%
		combined (complete object)	**41 943 040**	**+1 747.45%**
8^3	64 *mm*	Silhouette (outer cone)	22 020 096	+164.11%
		Structured Light (inner cone)	1 048 576	-82.83%
		combined (complete object)	**20 971 520**	**+823.72%**
16^3	32 *mm*	Silhouette (outer cone)	14 024 704	+68.21%
		Structured Light (inner cone)	3 276 800	-46.35%
		combined (complete object)	**10 747 904**	**+373.41%**
32^3	16 *mm*	Silhouette (outer cone)	10 649 600	+27.73%
		Structured Light (inner cone)	4 620 288	-24.35%
		combined (complete object)	**6 029 312**	**+165.57%**
64^3	8 *mm*	Silhouette (outer cone)	9 228 288	+10.68%
		Structured Light (inner cone)	5 367 808	-12.11%
		combined (complete object)	**3 860 480**	**+70.04%**
128^3	4 *mm*	Silhouette (outer cone)	8 676 352	+4.06%
		Structured Light (inner cone)	5 808 384	-4.89%
		combined (complete object)	**2 867 968**	**+26.32%**
256^3	2 *mm*	Silhouette (outer cone)	8 366 592	+0.34%
		Structured Light (inner cone)	6 026 208	-1.33%
		combined (complete object)	**2 340 384**	**+3.09%**
512^3	1 *mm*	Silhouette (outer cone)	8 335 984	-0.02%
		Structured Light (inner cone)	6 040 760	-1.09%
		combined (complete object)	**2 295 224**	**+1.10%**

Table 7.6: Reconstruction of synthetic cone with increasing octree resolution

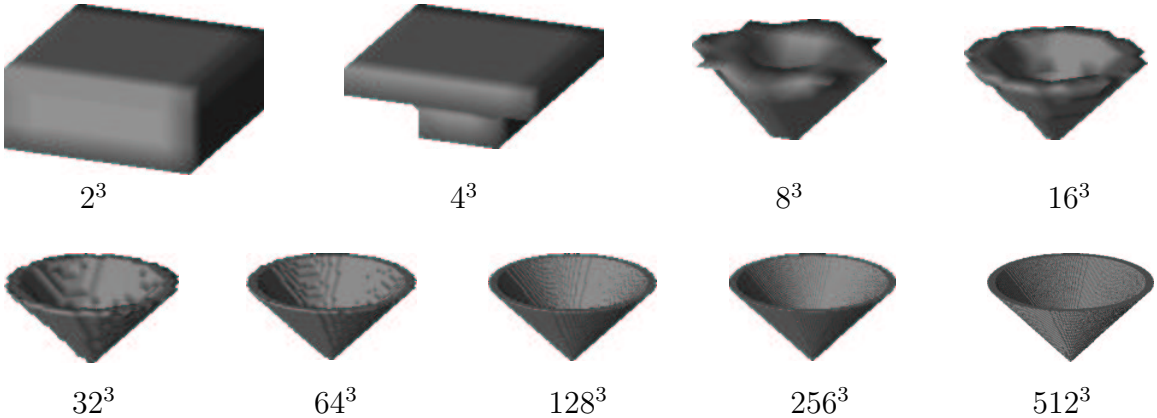


Figure 7.7: 3D models of synthetic cone with increasing octree resolution

views	*method*	*volume* (mm^3)	*error 1*	*error 2*
—	analytic (outer cone)	8 337 580	—	—
	analytic (inner cone)	6 107 256	—	—
	analytic (complete object)	**2 270 324**	—	—
360	Silhouette (outer cone)	8 366 592	+0.34%	—
	Structured Light (inner cone)	6 026 208	-1.33%	—
	combined (complete object)	**2 340 384**	**+3.09%**	—
4	Silhouette (outer cone)	9 798 464	+17.52%	+17.11%
	Structured Light (inner cone)	7 706 176	+26.18%	+27.88%
	combined (complete object)	**2 092 288**	**-7.84%**	**-10.60%**
10	Silhouette (outer cone)	8 507 120	+2.03%	+1.68%
	Structured Light (inner cone)	6 229 168	+2.00%	+3.37%
	combined (complete object)	**2 277 952**	**+0.34%**	**-2.67%**
20	Silhouette (outer cone)	8 433 184	+1.15%	+0.80%
	Structured Light (inner cone)	6 068 480	-0.63%	+0.70%
	combined (complete object)	**2 364 704**	**+4.16%**	**+1.04%**
30	Silhouette (outer cone)	8 415 760	+0.94%	+0.59%
	Structured Light (inner cone)	6 046 512	-0.99%	+0.34%
	combined (complete object)	**2 369 248**	**+4.36%**	**+1.23%**
60	Silhouette (outer cone)	8 393 952	+0.68%	+0.33%
	Structured Light (inner cone)	6 030 976	-1.25%	+0.08%
	combined (complete object)	**2 362 976**	**+4.08%**	**+0.97%**
90	Silhouette (outer cone)	8 383 424	+0.55%	+0.20%
	Structured Light (inner cone)	6 028 128	-1.30%	+0.03%
	combined (complete object)	**2 355 296**	**+3.74%**	**+0.64%**
180	Silhouette (outer cone)	8 371 808	+0.41%	+0.06%
	Structured Light (inner cone)	6 026 720	-1.32%	+0.01%
	combined (complete object)	**2 345 088**	**+3.29%**	**+0.20%**

Table 7.7: Reconstruction of synthetic cone with increasing number of views

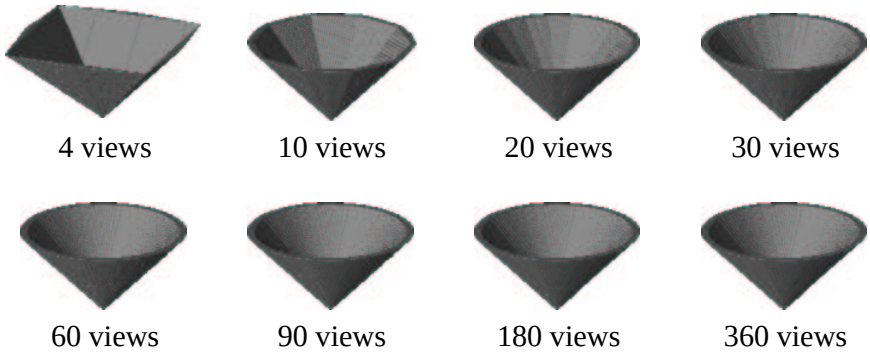


Figure 7.8: 3D models of synthetic cone with increasing number of views

The results are summarized in Table 7.9 and the constructed models shown in Figure 7.12.

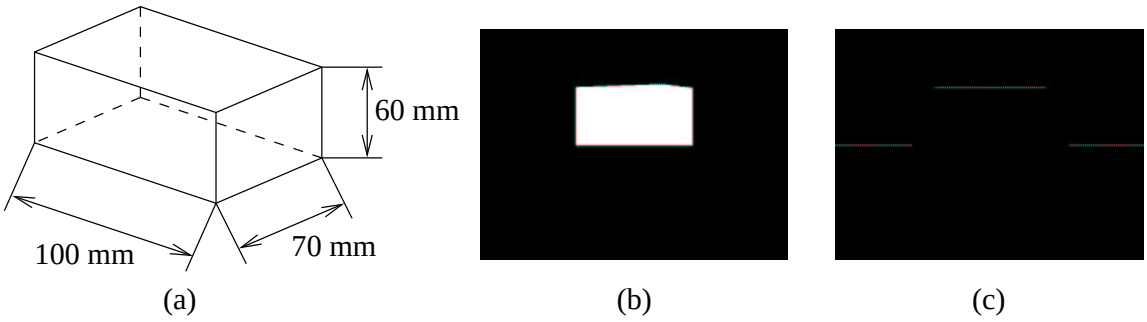


Figure 7.9: Synthetic cuboid (a) and an input image for Shape from Silhouette (b) and Shape from Structured Light (c)

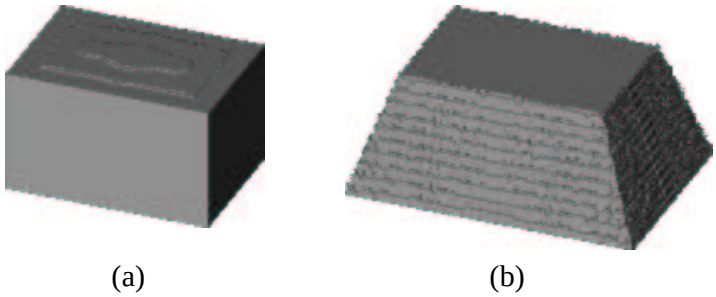


Figure 7.10: Models of the synthetic cuboid using either Shape from Silhouette (a) or Shape from Structured Light only (b)

octree	*voxel size*	*method*	*dimensions (mm)*	*volume (mm^3)*	*error*
—	—	analytic	$100.0 \times 70.0 \times 60.0$	420 000	—
2^3	256 *mm*	combined	$128.0 \times 128.0 \times 64.0$	1 048 576	+149.66%
4^3	128 *mm*	combined	$128.0 \times 128.0 \times 64.0$	1 048 576	+149.66%
8^3	64 *mm*	combined	$128.0 \times 96.0 \times 64.0$	786 432	+87.25%
16^3	32 *mm*	combined	$112.0 \times 80.0 \times 64.0$	573 440	+36.53%
32^3	4 *mm*	combined	$104.0 \times 72.0 \times 60.0$	449 280	+6.97%
64^3	2 *mm*	combined	$100.0 \times 72.0 \times 60.0$	432 000	+2.86%
128^3	1 *mm*	combined	$100.0 \times 70.0 \times 60.0$	420 000	0.00%

Table 7.8: Reconstruction of synthetic cuboid with increasing octree resolution

7.3.4 Analysis of Results

In the tests with synthetic sphere we compared Shape from Silhouette and Shape from Structured Light against one another. With respect to octree resolution there was no significant difference in the behavior of the two methods – the accuracy of the models built was approximately the same (see Table 7.4), with exception of octrees 256^3 and 512^3, where the volume and size of the Shape from Silhouette model started being smaller than the analytical values, while the Shape from Structured Light model truly converged to the analytical one. This can be explained through the complexity of building a Shape from

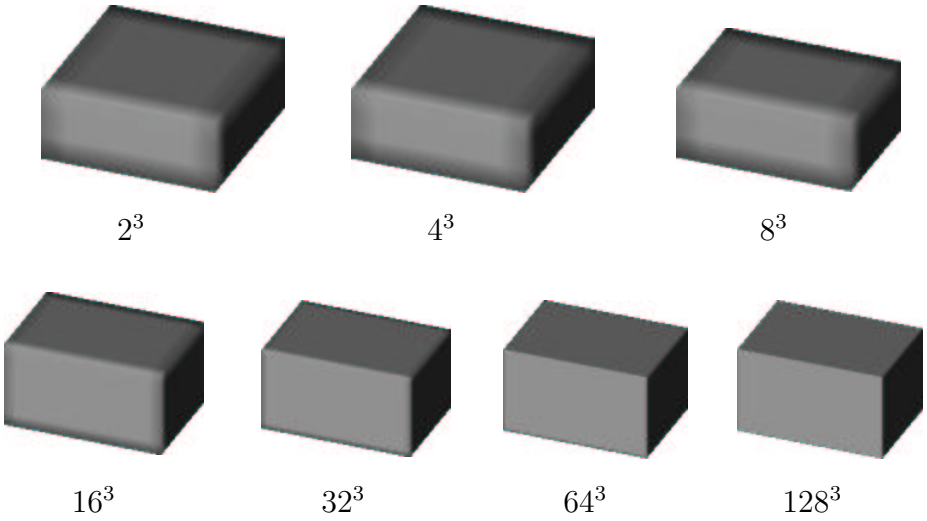


Figure 7.11: 3D models of synthetic cuboid with increasing octree resolution

views	*method*	*dimensions (mm)*	*volume (mm^3)*	*error*
—	analytic	$100.0 \times 70.0 \times 60.0$	420 000	—
4	combined	$104.0 \times 73.0 \times 60.0$	435 420	+3.67%
10	combined	$104.0 \times 97.5 \times 60.0$	508 746	+21.13%
20	combined	$104.0 \times 73.0 \times 60.0$	435 402	+3.67%
30	combined	$104.0 \times 76.0 \times 60.0$	446 304	+6.26%
60	combined	$104.0 \times 73.0 \times 60.0$	435 137	+3.60%
90	combined	$100.0 \times 73.0 \times 60.0$	427 891	+1.88%
180	combined	$100.0 \times 72.0 \times 60.0$	426 071	+1.45%
360	combined	$100.0 \times 70.0 \times 60.0$	420 000	0.00%

Table 7.9: Reconstruction of synthetic cuboid with increasing number of views

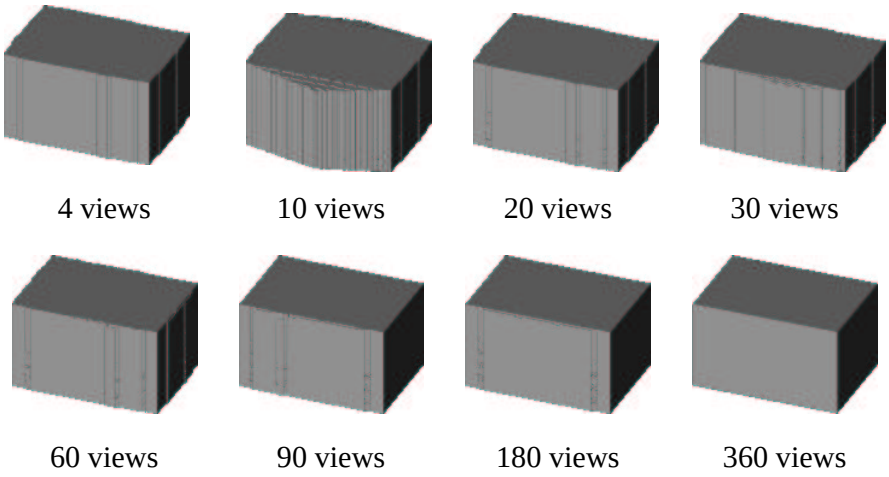


Figure 7.12: 3D models of synthetic cuboid with increasing number of views

Silhouette based octree – each node is projected into all 360 input images by projecting its 8 vertices, which means 2880 world-to-image projections of points per node. In the Shape from Structured Light method, a node is projected into two nearest images only, i.e., there are 16 world-to-image projections per octree node. Therefore, when dealing with octree nodes of finer resolution (when the projection of the node has approximately the size of a pixel), errors due to numerical instabilities are more likely to happen in Shape from Silhouette, especially when using a large number of views.

Regarding number of views, there was also no significant difference between the two methods. Using 20 instead of 360 input views was sufficient for both methods to create models less than 1% different from the models built using 360 views (see Table 7.5).

When building the synthetic cone the relative error of the computed volume of models built with increasing octree resolution (Table 7.6) was much larger than the error of the models of the sphere built with same parameters. The reason for this is that the cone has a large number of octree nodes belonging to the surface, and these nodes are the ones contributing mostly to the error.

The models of the cone built with different number of views showed the same behavior as the models of the sphere – starting from 20 input views the volume error relative to the volume of the object built with 360 views falls to 1% or less (Table 7.7).

The synthetic cuboid was the only object reconstructed perfectly, which can be explained by a perfect alignment of the object with the voxel space. However, the model was ideal only when using all 360 created views, both for Shape from Silhouette and Shape from Structured Light. When using any smaller number of views, the error was relatively large compared to the error of models of the sphere and the cone with the same number of views. This shows the complexity of reconstruction of flat surfaces. Interestingly, the model built using 4 views was better than the models built using 10 or 30 views. This illustrates the importance of selection of most significant views when reconstructing objects with flat surfaces using a small number of views.

If an object only needs to be visualized, without calculating its volume, a model built with the octree resolution 32^3 and 10 input views can give a satisfactory result.

However, it should be noted that the sufficient octree resolution as well as the sufficient number of view depend on the properties of the camera, the geometry of the acquisition system and the properties of the object observed. In the tests with synthetic data we dealt with relatively simple objects only. For more realistic cases with more complex data sets tests with real objects are necessary.

7.4 Real Objects

For tests with real objects we use 8 objects shown in Figure 7.13: a metal cuboid, a wooden cone, a globe, a coffee cup, two archaeological vessels and two archaeological sherds. The

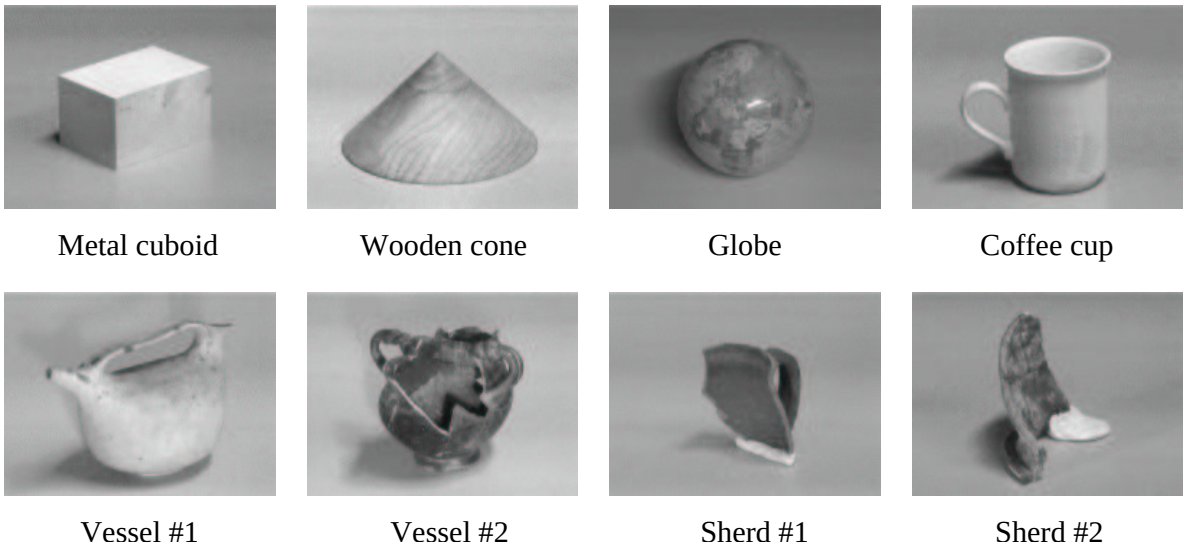


Figure 7.13: Real objects used for tests

cuboid, the cone and the globe have known dimensions so we can calculate their volumes analytically and compare them with the volumes of their reconstructed models. Using these three objects we can also measure the impact of ignoring camera lens distortion on the accuracy of the models. The other objects have unknown volume, so we will just show the models constructed. All models shown in this section are built using 360 views, with constant angle of 1° between two neighboring views.

7.4.1 Cuboid, Cone and Globe

For these three objects, we build models with octree resolution of 64^3, 128^3, 256^3 and 512^3, once with and once without taking lens distortion into account. Table 7.10 summarizes the results. Figure 7.14 shows the models built taking lens distortion into account.

7.4.2 Vessels, Sherds and Cup

As noted earlier in this section, the exact volumes of these objects are unknown and therefore the accuracy of the volume calculated through reconstruction can not be estimated. However, we can measure the bounding cuboid (i.e., the minimal possible cuboid aligned with the x, y and z axis of the world coordinate system which contains the object entirely) and compare it with the dimensions of the model. Table 7.11 summarizes the results. The resulting models, shown from three views, are depicted in Figure 7.15. All

object	*octree*	*lens d.*	*dimensions (mm)*	*volume (mm^3)*	*error*
cuboid (analytic)	—	—	$100.0 \times 70.0 \times 60.0$	420 000	—
cuboid	64^3	no	$102.0 \times 72.0 \times 60.0$	410 912	-2.16%
		yes	$102.0 \times 72.0 \times 60.0$	413 128	-1.64%
	128^3	no	$101.0 \times 71.0 \times 59.0$	381 922	-9.07%
		yes	$101.0 \times 71.0 \times 60.0$	387 018	-7.86%
	256^3	no	$101.0 \times 71.0 \times 59.0$	379 727	-9.59%
		yes	$101.0 \times 71.0 \times 60.0$	384 678	-8.41%
	512^3	no	$101.0 \times 71.0 \times 59.0$	379 727	-9.59%
		yes	$101.0 \times 71.0 \times 60.0$	384 678	-8.41%
cone (analytic)	—	—	$156.0 \times 156.0 \times 78.0$	496 950	—
cone	64^3	no	$153.0 \times 153.0 \times 79.2$	513 271	+3.28%
		yes	$152.2 \times 155.0 \times 77.5$	522 252	+5.09%
	128^3	no	$148.9 \times 150.2 \times 76.5$	457 833	-7.87%
		yes	$150.8 \times 150.8 \times 77.5$	463 526	-6.73%
	256^3	no	$147.5 \times 148.9 \times 75.8$	430 709	-13.33%
		yes	$150.1 \times 149.4 \times 77.5$	435 180	-12.43%
	512^3	no	$147.5 \times 148.9 \times 75.8$	428 930	-13.69%
		yes	$148.0 \times 149.4 \times 77.5$	433 067	-12.86%
globe (analytic)	—	—	$149.7 \times 149.7 \times 149.7$	1 756 564	—
globe	64^3	no	$151.0 \times 151.0 \times 147.3$	1 851 896	+5.43%
		yes	$151.0 \times 154.5 \times 147.3$	1 867 583	+6.32%
	128^3	no	$149.1 \times 149.1 \times 145.6$	1 752 341	-0.24%
		yes	$151.0 \times 149.1 \times 145.6$	1 766 591	+0.57%
	256^3	no	$148.2 \times 148.2 \times 144.6$	1 703 854	-3.00%
		yes	$149.1 \times 148.2 \times 144.6$	1 717 624	-2.22%
	512^3	no	$148.2 \times 148.2 \times 144.6$	1 697 687	-3.35%
		yes	$149.1 \times 148.2 \times 144.6$	1 711 564	-2.56%

Table 7.10: Reconstruction of cuboid, cone and globe with different octree resolutions

models are built with an octree resolution of 256^3 and using 360 views.

object	*voxel size*	*measured dimensions (mm)*	*calculated dimensions (mm)*	*volume (mm^3)*
vessel #1	0.74 *mm*	$141.2 \times 84.8 \times 93.7$	$139.2 \times 83.2 \times 91.4$	336 131
vessel #2	0.53 *mm*	$114.2 \times 114.6 \times 87.4$	$113.0 \times 111.9 \times 86.4$	263 696
sherd #1	0.84 *mm*	$51.8 \times 67.0 \times 82.2$	$51.0 \times 66.0 \times 79.4$	35 911
sherd #2	0.76 *mm*	$76.0 \times 107.3 \times 88.5$	$74.9 \times 103.9 \times 86.2$	38 586
cup	0.66 *mm*	$113.3 \times 80.0 \times 98.9$	$111.6 \times 79.0 \times 98.3$	276 440

Table 7.11: Reconstruction of two vessels, two sherds and a cup

In order to illustrate how much each of the algorithms (Shape from Silhouette and Shape from Structured Light) contributes to the final models from Figure 7.15, Figure 7.16 shows models using one of the methods only, as well as using the combination of both.

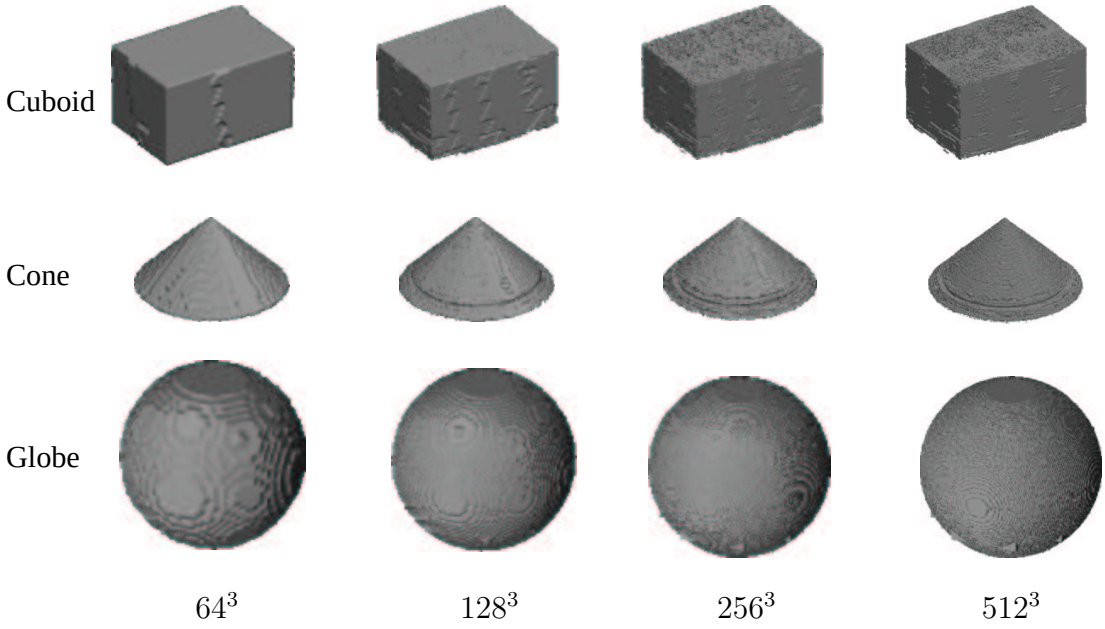


Figure 7.14: 3D models of cuboid, cone and globe with different octree resolutions

7.4.3 Analysis of Results

The tests with real objects showed the difference between using a theoretical and a real-world acquisition system. The errors in computed dimensions and volume were by an order of magnitude larger than the errors with synthetic data. However, the errors were consistent — the computed volume for fine grained models (256^3 and 512^3) was always smaller (from 3% to 13%) than the analytical volume of the three objects with known volume (see Table 7.10) and the size of the bounding cuboid was usually 1–5 mm smaller than the real size in each dimension (see Tables 7.10 and 7.11). Exception were models of the cone whose dimensions in x and y direction were up to 8 mm smaller than the real dimensions (see Table 7.10).

The main reason for these inaccuracies turned out to be the input images, especially the process of their binarization. The binarization of both silhouette and laser images (described in Sections 5.2.1 and 5.2.2) is partly based on automatic thresholding of the acquired images and for some images the thresholding interpreted object pixels close to the object border as background, resulting in models smaller than the corresponding objects. This problem was especially present for the side of the objects bordering the turntable, which explains why the error was the largest for the cone and the smallest for the globe (see Table 7.10). The cone has a large base leaning on the turntable, while the globe only touches the turntable in an almost tangential way.
Another problem related to the image acquisition itself was in the occasional camera synchronization errors — by acquiring a large number of images (720 images per object, at rate of one image in about 2.5 seconds) it was impossible to manually proof every single image — there were several images which contained some pixels from the previously

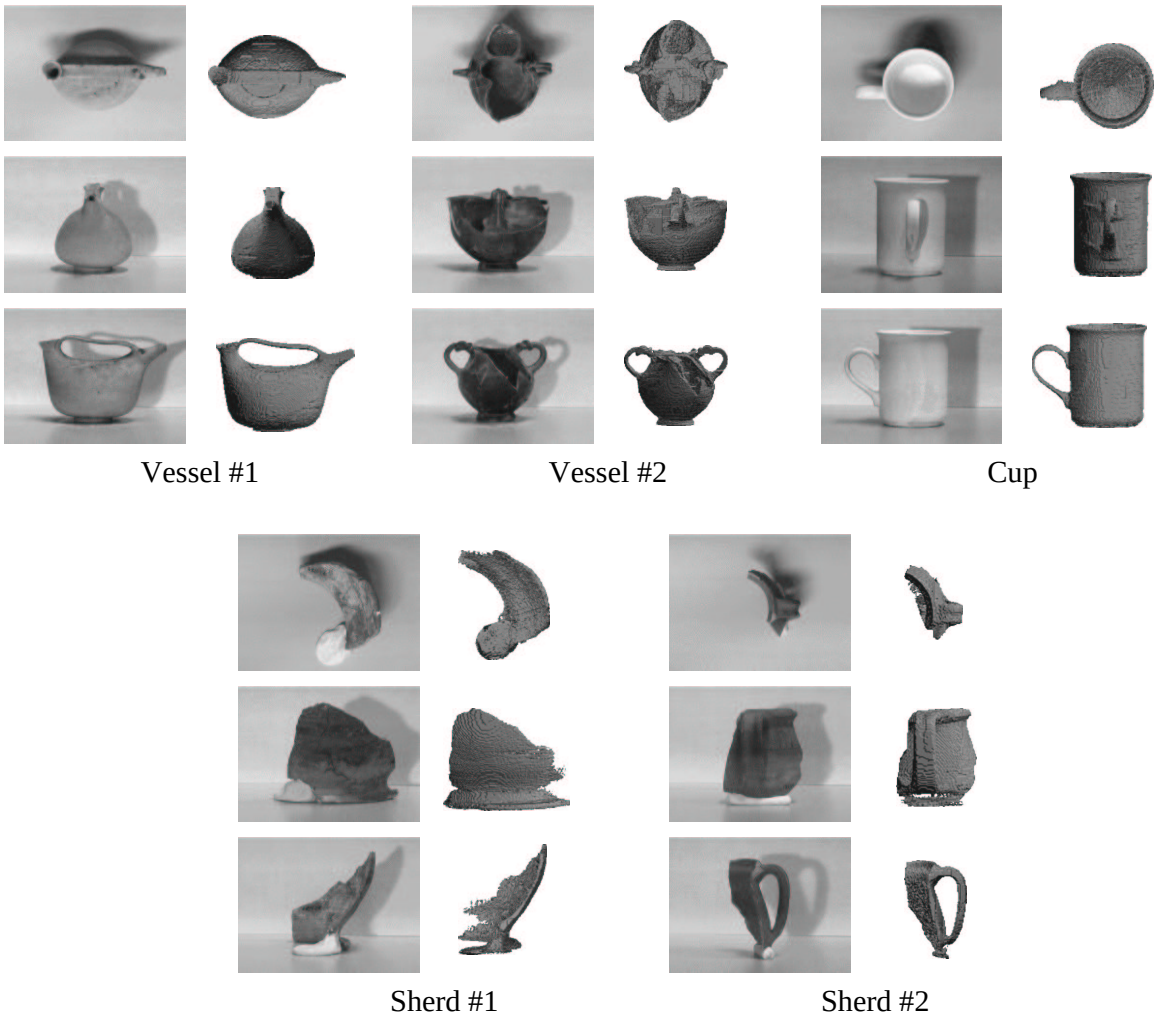


Figure 7.15: 3D models of two vessels, two sherds and a cup

acquired image. This especially played a role in the Shape from Structured Light part, because having a white pixel on a wrong place in the image could cause the algorithm to simply cut off a part of the object. For example, the right handle of the vessel #2 in Figure 7.16, built using the combined method, was partly cut off because of this reason. Regarding lens distortion, Table 7.10 indicates that the models built taking lens distortion into account were always slightly better than the models built ignoring it, but there was no significant difference, which can be expected when the objects stay mainly close to the center in all input images.

Purely visually, all the models built using the combined method looked truthful (see Figures 7.15 and 7.16) and except the inside of the vessel #2 and the cup, the objects were reconstructed completely, with all their cavities and concavities.

Comparing models built using one of the methods only, either Shape from Silhouette or Shape from Structured Light (see Figure 7.16) we can see than Shape from Silhouette

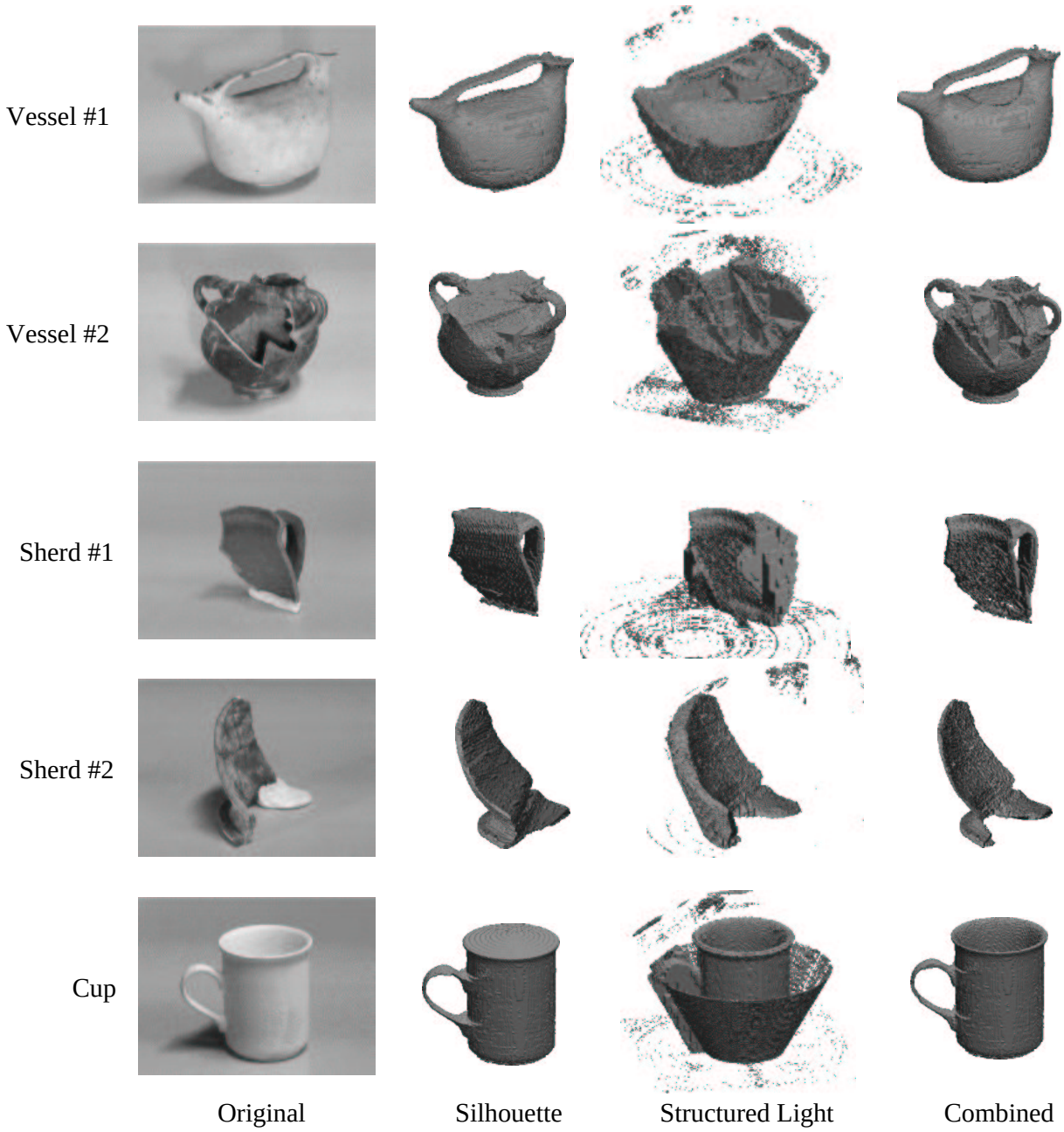


Figure 7.16: Comparison of models built with Shape from Silhouette, Shape from Structured Light and the combined method

alone is able to reconstruct the convex hull of an object in an accurate way. Shape from Structured Light alone, in the form presented in this work, does not create usable models. However, it is a great method to supplement the weaknesses of Shape from Silhouette and recover the cavities not visible in silhouette images.

7.5 Next View Planning

For the synthetic cuboid from Figure 7.9 and for all real objects from Figure 7.13 we also built models based on views obtained through Next View Planning. As described in Section 6.3.3, user definable parameters for NVP are the maximal and the initial step between two neighboring views, as well as the maximal allowed difference between them. The parameter with the greatest impact on the number of the views selected is the difference between two images. We experimented with different values of this parameter and it turned out that the most reasonable value is dependent on the complexity of the surface of the object modeled. For all objects presented the most reasonable value was in the range from 2–15%, both for Shape from Silhouette and Shape from Structured Light. It was low for highly symmetrical objects (the cuboids and the cone) and high when the object was not placed in the center of the turntable (the two sherds). For all objects the maximal step was set to 16° and the initial to 4° for silhouette and 2° for laser images.

In order to evaluate the NVP-based models, we compare them with models built with nearly the same number of equiangular views and with models built using all 360 possible views. We expect to see that the volume of NVP-based models is closer to the volume of models built using all views than the models built with equiangular views. Figure 7.17 shows the models built and Table 7.12 summarizes the results. All models were built using octree resolution of 256^3 and taking into account the lens distortion. The column labeled "#silh." in Table 7.12 refers to the number of Shape from Silhouette views and "#laser" to the number of Shape from Structured Light views used.

The results in Table 7.12 indicate that for none of the objects there is a significant difference between the volume computed using NVP-based and equiangular views. This can be expected for objects with asymmetric, highly detailed surface, such as the vessels and the sherds, or completely rotationally symmetric objects, such as the cone or the globe. For simply shaped, but asymmetrical objects, such as the cuboids and the cup, a certain increase in the accuracy of the models built using NVP could be expected. In order to additionally examine our NVP algorithm, in Figure 7.19 we illustrate the views selected for the synthetic and real cuboid, the cone and the cup. All objects are showing the objects from the top view, facing the x-y plane of the world coordinate system (shown in Figure 3.3).

For Shape from Silhouette views (Figures 7.19a, 7.19c, 7.19e and 7.19g) each dashed line indicates the direction the camera was viewing from, i.e., it represents the camera's optical axis. High density scanning areas should be those for which the silhouette border moves fast, e.g., when the width of the silhouette changes rapidly. This happens when an object's part which is far from the rotational axis starts or ends being visible from the camera. For the cuboids (Figures 7.19a and 7.19c) such parts are its corners, for the cone (Figure 7.19e) there are no such parts and for the cup (Figure 7.19g) it is its handle. For the purpose of better understanding of the selected Shape from Silhouette views in Figure 7.19, Figure 7.18 illustrates the difference between two views. Also in this figure, each dashed line represents the optical axis of the camera. If we define O from Figure 7.18

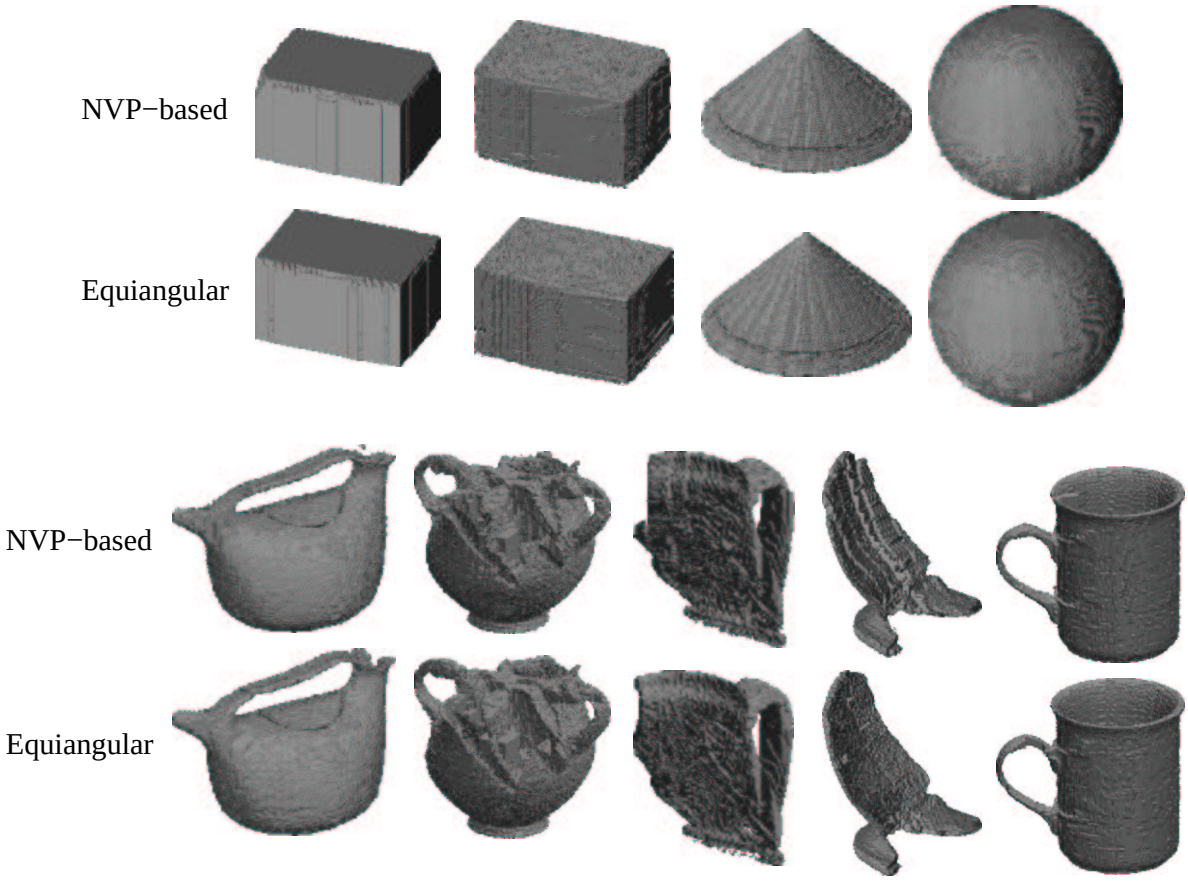


Figure 7.17: Comparison of models built using NVP-based and equiangular views

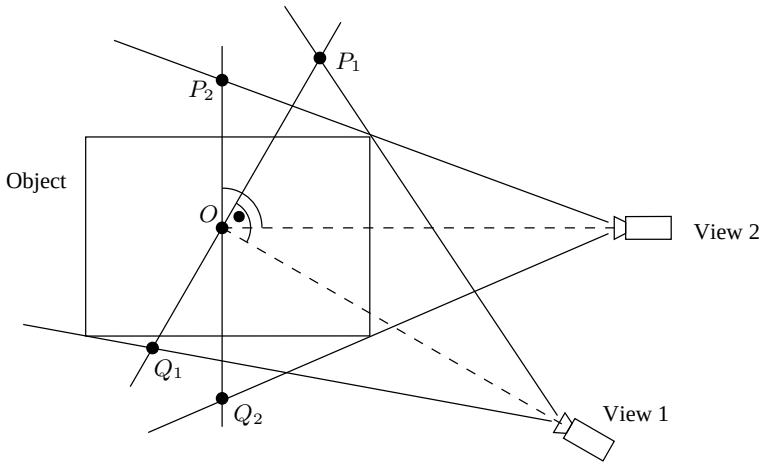


Figure 7.18: Difference between two silhouette views

as the point representing the rotational axis of the turntable, then we can view the lines $\overline{P_1Q_1}$ and $\overline{P_2Q_2}$ as the width of the silhouette in views 1 and 2, respectively. The parts of an object for which this width changes significantly from one view to the next need to

object	*view selection*	*#silh.*	*#laser*	*dimensions (mm)*	*volume (mm^3)*	*error*
synth. cuboid	all	360	360	$100.0 \times 70.0 \times 60.0$	420 000	—
	NVP-based	54	71	$103.5 \times 74.0 \times 60.0$	436 666	+3.97%
	equiangular	60	72	$104.0 \times 73.0 \times 60.0$	434 248	+3.39%
real cuboid	all	360	360	$101.0 \times 71.0 \times 60.0$	384 678	—
	NVP-based	54	79	$101.6 \times 72.3 \times 60.0$	397 937	+3.45%
	equiangular	60	90	$101.6 \times 71.9 \times 59.5$	397 684	+3.38%
cone	all	360	360	$150.1 \times 149.4 \times 77.5$	435 180	—
	NVP-based	24	25	$151.6 \times 151.6 \times 76.5$	462 155	+6.20%
	equiangular	24	24	$151.6 \times 152.2 \times 76.5$	462 207	+6.21%
globe	all	360	360	$149.1 \times 148.2 \times 144.6$	1 717 624	—
	NVP-based	24	25	$150.0 \times 149.1 \times 144.6$	1 733 613	+0.93%
	equiangular	24	24	$150.0 \times 150.0 \times 144.6$	1 732 919	+0.89%
vessel #1	all	360	360	$139.2 \times 83.2 \times 91.4$	336 131	—
	NVP-based	52	45	$138.5 \times 84.0 \times 92.1$	341 216	+1.51%
	equiangular	52	45	$139.2 \times 83.2 \times 92.1$	339 227	+0.92%
vessel #2	all	360	360	$112.9 \times 111.8 \times 86.4$	263 696	—
	NVP-based	55	148	$113.4 \times 111.8 \times 86.3$	269 321	+2.13%
	equiangular	60	120	$113.4 \times 112.3 \times 86.3$	269 819	+2.32%
sherd #1	all	360	360	$51.0 \times 66.0 \times 79.4$	35 911	—
	NVP-based	99	163	$50.1 \times 66.0 \times 80.1$	37 237	+3.69%
	equiangular	90	180	$50.9 \times 61.8 \times 80.2$	38 047	+5.95%
sherd #2	all	360	360	$74.8 \times 103.9 \times 86.3$	38 586	—
	NVP-based	90	181	$74.8 \times 103.8 \times 87.0$	40 751	+5.61%
	equiangular	90	180	$74.8 \times 103.8 \times 87.0$	41 565	+7.72%
cup	all	360	360	$111.6 \times 79.0 \times 98.3$	276 440	—
	NVP-based	36	34	$112.2 \times 80.4 \times 100.3$	286 265	+3.55%
	equiangular	36	36	$112.2 \times 79.7 \times 102.3$	282 989	+2.37%

Table 7.12: Comparison of models built using all views, NVP-based views and equiangular views

be scanned with a higher density.

Shape from Structured Light views (Figures 7.19b, 7.19d, 7.19f and 7.19h) are easier to understand. A dashed line in these figures represents the laser plane projected onto the object for that view. High density scanning areas should be those where the distance between the rotational axis and the intersection point of the laser plane and the object surface changes rapidly. For the cuboids (Figures 7.19b and 7.19d) such areas are the ones around the cuboids' corners, for the cone (Figure 7.19f) these areas do not exist, and for the cup (Figure 7.19h) they lie around the cup handle.

Let us analyze each of the objects from Figure 7.19. For the silhouette views of the cuboids (Figures 7.19a and 7.19c) the views with the highest density are 0°–60° and 180°–240°. That makes sense, because the width of the cuboid silhouettes as defined in Figure 7.18 is smallest for views from 30° and 210° and largest from approximately 75°, 165°, 255° and 345°. For views close to 30° and 210° the silhouette width is determined by the two corners close to the camera (see View 2 in Figure 7.18). Because of being close to the camera these corners move almost orthogonally as the turntable moves, so the

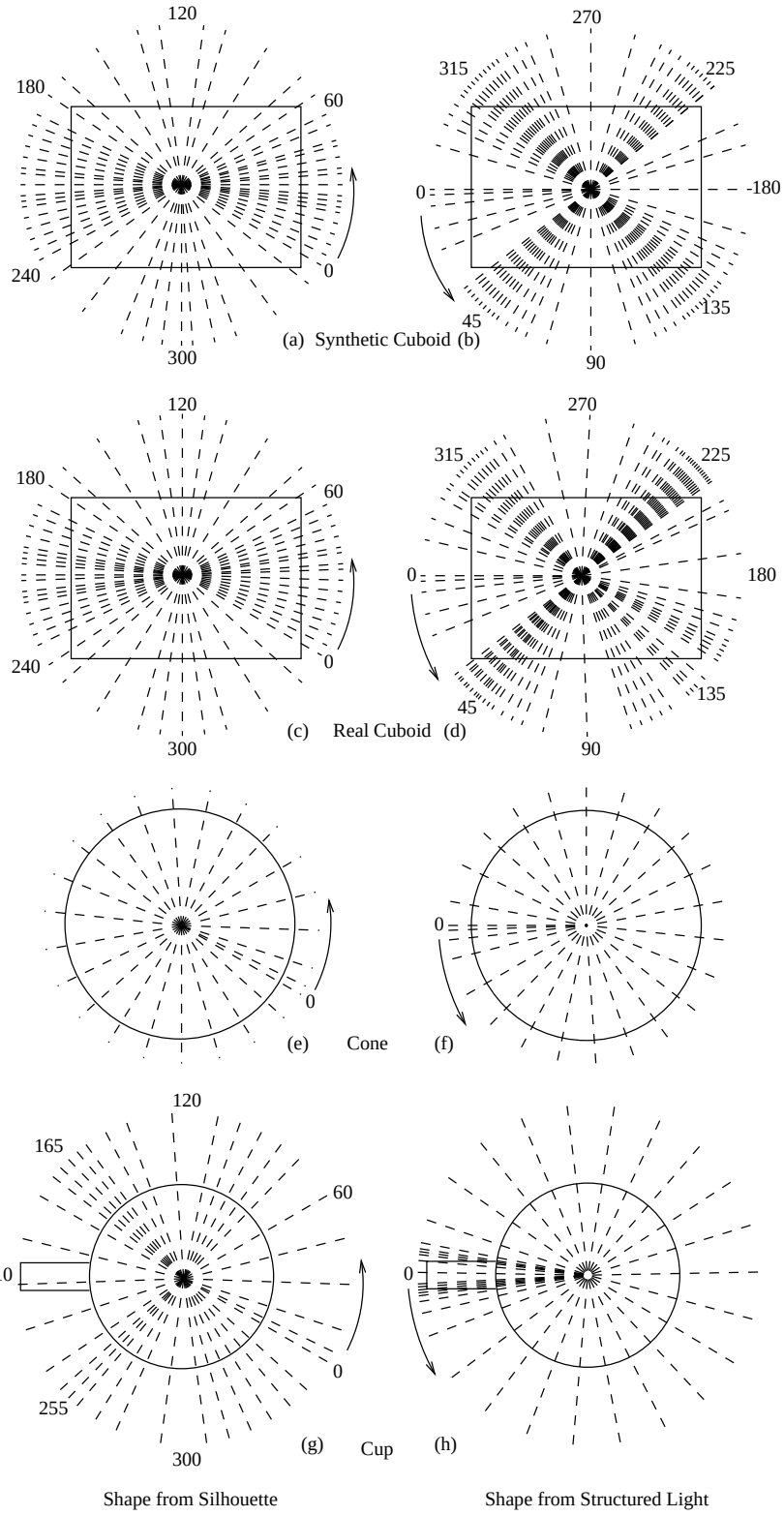


Figure 7.19: Analysis of selected views for cuboids, cone and cup

silhouette width changes rapidly here and the scans are most dense in these areas. The laser views of the cuboids (Figure 7.19b and 7.19d) are more dense close to the corners, as expected, but it can also be seen that for both synthetic and the real cuboid several corners were missed, because the step was too large, but the NVP algorithm did not see it — for example, the next left and next right view of the lower right corner intersect the cuboid surface at almost the same distance from the rotational center.

For both silhouette and laser views of the cone (Figures 7.19e and 7.19f) not much needs to be said — all views look nearly the same, so the step between two views was constantly equal to the maximal allowed step. The step was smaller only for views close to 0°, solely because of the starting angle being smaller than the maximal angle.

For the silhouette views of the cup (Figure 7.19g) high density view were taken from angles close to 165° and 255°. This is expected, because for those views the cup handle starts/ends being visible (i.e., not occluded by the body of the cup). The laser views (Figure 7.19h) are dense only in areas where the projected laser plane "jumps" from the body of the cup to its handle and back, just as expected.

Obviously, our NVP algorithm did not fail in choosing the "right" views (except for the laser views of corners of the cuboids), but it did not bring any significant improvements in the results (measured in terms of the volume and the size of the objects) compared to the models built using an equivalent number of equiangular views. We believe that the reason lies in the errors from the image acquisition and binarization process (due to camera synchronization errors and automatic thresholding) — that they were significantly larger than what could be made up by NVP-based selection of views. For example, the NVP-based model of the cup in Figure 7.17 contains the complete handle, whereas the model built using equiangular views misses some parts close to the top of the handle — they were cut off through irregularities in input images. On the other hand, the NVP-based model also contains some non-existing volume inside the cup, close to the handle, also caused by errors in the input images.

7.6 Performance Issues

This section gives an idea how much time it takes to perform all steps of the modeling algorithm presented. The process between having an uncalibrated acquisition system and producing a 3D model of an object, in terms of the tasks performed, can be observed through 4 steps:

1. Calibration of the acquisition system. This step, described in detail in Chapter 3, requires a lot of manual work and takes about 30–60 minutes to complete.

2. Image acquisition. On our system, the computer connected to the camera was a 233 MHz Pentium with 256 MB of RAM. The automatic acquisition of all 360 possible views took about 13–15 minutes, i.e., one image in 2.5 seconds. NVP-based acquisition requires additional image processing during the acquisition process and,

depending on the complexity of the object acquired, it took between 3 and 45 minutes to complete.

3. Binarization of input images, as described in Section 5.2. For this and the next step, we used a 450 MHz Pentium II with 128 MB or RAM. For silhouette images, the binarization of 360 images took about 34 seconds. The binarization of laser images, being more complex, took between 9 and 12 minutes, depending on the complexity of input images.

4. The actual model building. Depending on the maximal octree resolution and the number of views used, as well as on the other user-definable parameters, this process took anywhere between 8 seconds and 60 minutes for the objects presented in this section. For large number of views Shape from Structured Light was by an order of magnitude faster than Shape from Silhouette, in some instances up to 15 times faster using the same number of views and the same octree resolution.

The times needed for the step 4, model building, are elaborated in Table 7.13 for synthetic, and in Table 7.14 for real objects. The column "#silh." refers to the number of Shape from Silhouette, and "#laser" to the number of Shape from Structured Light views of the object. Accordingly, "t_{CPU} silh." refers to the CPU time needed for building the model using Shape from Silhouette only, "t_{CPU} laser" to the time for Shape from Structured Light based models, and "t_{CPU} comb." to the time to build the models using the combined method. All CPU times are given in seconds.

From the tables 7.13 and 7.14 we can draw several conclusions:

- The time needed for completion of both Shape from Silhouette and Shape from Structured Light linearly grows with an increasing octree resolution.
- The time needed for Shape from Silhouette is also linearly dependent on the number of views, whereas for Shape from Structured Light the time is nearly constant for any number of views.
- Shape from Structured Light is by an order of magnitude faster than Shape from Silhouette for the same octree resolution and the same number of views.
- Ignoring camera lens distortion reduces the time needed to build a model by 75%.
- Ignoring the lens distortion while reducing the number of silhouette images to 30 or 40 at the same time, a performance gain of more than 95% is possible without heavily penalizing the accuracy of the final model.

7.7 Summary

This chapter presented the experimental results of the 3D modeling method described in this work and gave their analysis. Experiments with synthetic objects were performed in order to evaluate the modeling algorithm without any influence of camera calibration

object	*octree*	*#silh.*	*#laser*	t_{CPU} *silh.*	t_{CPU} *laser*	t_{CPU} *comb.*
sphere	2^3	360	360	16.21	8.41	24.65
	4^3	360	360	26.58	8.56	35.08
	8^3	360	360	43.85	8.52	51.83
	16^3	360	360	63.65	9.10	72.51
	32^3	360	360	91.22	9.70	100.69
	64^3	360	360	136.99	11.75	149.11
	128^3	360	360	255.66	17.68	274.74
	256^3	360	360	540.92	40.61	581.44
	512^3	360	360	981.45	91.64	1031.05
	256^3	4	4	15.81	14.44	20.50
	256^3	10	10	25.66	13.61	29.21
	256^3	20	20	39.31	13.90	42.52
	256^3	30	30	52.86	14.40	55.24
	256^3	60	60	95.92	15.85	97.82
	256^3	90	90	135.11	17.58	140.67
	256^3	180	180	274.59	23.89	269.93
cone	2^3	360	360	9.71	N/A	18.44
	4^3	360	360	12.70	N/A	21.13
	8^3	360	360	16.79	N/A	24.86
	16^3	360	360	25.06	N/A	32.14
	32^3	360	360	33.71	N/A	42.45
	64^3	360	360	52.07	N/A	61.26
	128^3	360	360	99.62	N/A	141.87
	256^3	360	360	195.15	58.94	332.58
	512^3	360	360	361.38	105.39	611.73
	256^3	4	4	10.68	N/A	15.44
	256^3	10	10	14.01	N/A	20.77
	256^3	20	20	18.89	N/A	29.06
	256^3	30	30	23.82	N/A	38.92
	256^3	60	60	43.28	N/A	69.78
	256^3	90	90	55.42	N/A	91.72
	256^3	180	180	103.07	N/A	171.26
cuboid	2^3	360	360	N/A	N/A	17.04
	4^3	360	360	N/A	N/A	17.84
	8^3	360	360	N/A	N/A	20.04
	16^3	360	360	N/A	N/A	22.98
	32^3	360	360	N/A	N/A	28.18
	64^3	360	360	N/A	N/A	44.50
	128^3	360	360	N/A	N/A	72.78
	256^3	4	4	N/A	N/A	9.72
	256^3	10	10	N/A	N/A	12.92
	256^3	20	20	N/A	N/A	13.76
	256^3	30	30	N/A	N/A	16.80
	256^3	60	60	N/A	N/A	22.90
	256^3	90	90	N/A	N/A	22.50
	256^3	180	180	N/A	N/A	38.46
	256^3	54	71	N/A	N/A	21.88

Table 7.13: CPU times for reconstruction of synthetic objects

object	*octree*	*#silh.*	*#laser*	*CPU time*
cuboid	64^3	360	360	219.90
	128^3	360	360	699.18
	256^3	360	360	1638.04
	512^3	360	360	2109.74
	256^3	54	79	329.82
cuboid (lens distortion ignored)	64^3	360	360	69.17
	128^3	360	360	154.90
	256^3	360	360	393.64
	512^3	360	360	422.15
cone	64^3	360	360	181.33
	128^3	360	360	565.15
	256^3	360	360	1871.33
	512^3	360	360	2035.25
	256^3	24	25	153.73
cone (lens distortion ignored)	64^3	360	360	69.52
	128^3	360	360	142.26
	256^3	360	360	398.35
	512^3	360	360	451.21
globe	64^3	360	360	217.97
	128^3	360	360	615.68
	256^3	360	360	1782.76
	512^3	360	360	3995.65
	256^3	24	25	142.16
globe (lens distortion ignored)	64^3	360	360	95.47
	128^3	360	360	170.07
	256^3	360	360	401.25
	512^3	360	360	830.15
vessel #1	256^3	360	360	1313.70
	256^3	52	45	227.49
vessel #2	256^3	360	360	2345.72
	256^3	55	148	458.31
sherd #1	256^3	360	360	605.74
	256^3	99	163	174.68
sherd #2	256^3	360	360	832.67
	256^3	90	181	203.92
cup	256^3	360	360	1891.52
	256^3	36	34	197.91

Table 7.14: CPU times for reconstruction of real objects

errors and/or imperfect input images. These tests showed that it is possible to build models with the volume less than 1% different from the analytical value. Experiments with real objects showed the behavior of the modeling algorithm in a real environment. The models were visually truthful, but slightly (about 10%) smaller than the actual objects, due to hardware caused irregularities in input images and information loss in binarization of these images. Experiments with Next View Planning were also presented, and they indicated that our NVP method does choose "good" views, but it does not bring any significant improvement in the accuracy of the models produced, compared to the models built using the same number of equiangular views. Finally, the performance

of the modeling algorithm has also been analyzed, suggesting that the two parameters that can influence the speed of the algorithm the most are the camera lens distortion (i.e., whether it is ignored or not) and the number of Shape from Silhouette views.

Chapter 8

Conclusion and Outlook

The work presented in this thesis proposed a 3D modeling method based on combination of Shape from Silhouette and laser based Shape from Structured Light, using a turntable to obtain multiple views of an object. The intended application of the method is reconstruction of archaeological vessels and sherds, although it can be applied to any kind of objects.

The purpose of combining Shape from Silhouette and Shape from Structured Light was to create a method which will use the advantages and overcome the weaknesses of both underlying methods. The main advantage of Shape from Silhouette is that it can quickly obtain an approximate volumetric model of a complete object, including objects with handles, such as many archaeological vessels. Its main limitation is that it is not able to detect concavities on the surface of an object, what makes it incapable of reconstructing most of archaeological sherds. Shape from Structured Light, on the other hand, can theoretically reconstruct the complete surface of an object, including the concave surfaces, but the models built using Shape from Structured Light only are usually incomplete, due to camera and light occlusions.

Because of the different nature of these two methods (volumetric vs. surface model), the main problem that needed to be solved when combining them was to find a data structure which can be used by both methods. We decided to use a volumetric representation in the form of an octree, and build a model in an incremental way — a starting coarse model is refined in a level-by-level manner using all available silhouette and laser light images at each level. Doing so, large coherent blocks of the object are modeled quickly and the only higher level octree nodes processed are the ones close to the object's surface. When processed, each node of the octree is projected into all silhouette images and intersected with the object's silhouette, and it is also projected into the nearest laser plane and tested whether it lies below or above the intersection of that laser plane with the object's surface.

The experiments with synthetic objects showed that construction of nearly perfect models is possible, limited only by image and model resolution. In the experiments with real objects the results were less accurate — the computed volume was about 10% smaller than the actual object volume — this was mainly caused by errors in the input images and inaccuracies in their binarization, as a preprocessing step for the modeling algorithm.

However, the algorithm was able to produce complete and visually faithful models for all objects, including sherds and vessels with concave surfaces and a handle. Only the inside of deep objects could not be completely recovered, due to camera and light occlusions.

Using all possible views (360 silhouette and 360 laser light views) the modeling takes up to 60 minutes on a low-end hardware (450 MHz CPU and 128 MB RAM), but by reducing the number of views by 70–80% and ignoring the lens distortion at the same time, nearly the same models can be built in less than a minute on the same hardware. The number of views can be reduced either by using a fixed step of more than 1° between two views or by introducing Next View Planning, which selects the views based on features of the object modeled. As the secondary goal of this thesis, a simple NVP algorithm was developed, both for Shape from Silhouette and Shape from Structured Light. The algorithm performs NVP-based image acquisition by doubling or halving the step depending on the change it percepts between two most recently acquired images. The analysis of our NVP method showed that it does make intelligent choice of the input images, but in our tests it did not bring any significant improvement in the accuracy of the models produced when compared with models built using the same number of views based on a constant step between two views. However, it can be argued that any accuracy gain was outweighed by the errors in acquiring and processing the input images.

Overall, our combined modeling approach proved to be useful for automatic creation of virtual models of arbitrarily shaped objects. With respect to its archaeological application it can provide models of any kind of archaeological pottery for, for example, virtual museums. The models can also be intersected with arbitrary planes, resulting in profile sections of a sherd or a vessel. Furthermore, the volume of an object can be estimated, including the inside volume of objects such as bowls or cups. However, for high precision measurements of the volume our method did not produce sufficiently accurate results, but it gave a good rough estimate, which is sufficient for most archaeological applications.

There are several possibilities how our method could be improved:

- More reliable image acquisition. Because of camera synchronization errors it would make sense to acquire multiple images for each view and use the image which is least likely to contain errors.
- More robust binarization of acquired images. Binarization of both silhouette and laser light images employs automatic or user defined global thresholding of an image, which in many cases made the silhouette of an object smaller than it should be, resulting in models smaller than the objects. More sophisticated techniques, such as adaptive thresholding or edge detection should produce better results.
- More accurate testing of octree nodes against laser light images. An octree node is projected into the nearest plane only, also in cases when the node lies somewhere between two neighboring scanned planes or if it spans across several scanned planes. The decision whether the node belongs to the background or the object would be more accurate if these cases would also be considered.

- Improvement of the NVP algorithm. While simple and straight-forward, our NVP method showed not sophisticated enough for acquisition of laser light images, failing to scan all corners of the cuboid. A partial cause for this failure is that our method simply ignores unseen portions of the object when deciding which view to choose next. It would be better to give the unseen areas a special treatment. Furthermore, not being the main focus of the thesis, Next View Planning was not given the same attention as the modeling method — further evaluation of our NVP approach should be done with more complex synthetic data, in order to eliminate the impact of calibration and image acquisition errors on the results.

Improvements listed above would make the method significantly more accurate in terms of estimating the dimensions and the volume of objects. The method could also be combined with other *Shape-from-X* techniques, such as Shape from Stereo, Shape from Shading or Shape from Texture — the base for combining Shape from Silhouette and Shape from Structured Light in our approach was the common octree model, used by both methods, so possibilities could be investigated to use the octree model for other Shape-from-X methods. Another interesting enhancement would be to take additional color images of an object and perform texture mapping onto the model. This would broaden the range of applications of our 3D modeling method.

Bibliography

[AAK71] Y. I. Abdel-Aziz and H. M. Karara. Direct linear transformation into object space coordinates in close-range photogrammetry. In *Symphosium on Close-Range Photogrammetry*, pages 1–18, Urbana, Illinois, January 1971. University of Illinois at Urbana-Champaign.

[AKK+01] K. Adler, M. Kampel, R. Kastler, M. Penz, R. Sablatnig, K. Schindler, and S. Tosovic. Computer aided classification of ceramics — achievements and problems. In *Proceedings of 6th International Workshop on Archaeology and Computers*, Vienna, Austria, November 2001. in press.

[Bak77] H. Baker. Three-dimensional modelling. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 649–655, 1977.

[Bes88] P. J. Besl. Active, optical range image sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[BFB+98] N. A. Borghese, G. Ferrigno, G. Baroni, A. Pedotti, S. Ferrari, and R. Savarè. Autoscan: A flexible and portable 3D scanner. *IEEE Computer Graphics and Applications*, 18(3):38–41, 1998.

[BJ88] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:167–192, March 1988.

[BL00] A. Bottino and A. Laurentini. Non-intrusive silhouette based motion capture. In *Proceedings of 4th World Multiconference on Systemics, Cybernetics and Informatics*, pages 23–26, July 2000.

[BP98] J. Y. Bouguet and P. Perona. 3D photography on your desk. In *Proceedings of the 6th IEEE International Conference on Computer Vision*, pages 43–50, 1998.

[BV99] J. S. De Bonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 418–425, 1999.

[CA86] C. H. Chien and J. K. Aggarwal. Volume/surface octrees for the representation of three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 36:100–113, 1986.

[CH88] H. H. Chen and T. S. Huang. A survey of construction and manipulation of octrees. *Computer Vision, Graphics, and Image Processing*, 43:409–431, 1988.

[CK88] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:407–416, May 1988.

[Con85] C. Connolly. The determination of next best views. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 432–435, 1985.

[DC01] J. Davis and X. Chen. A laser range scanner designed for minimum calibration complexity. In *Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling*, pages 91–98, May 2001.

[DF99] Q. Delamarre and O. Faugeras. 3D atriculated models and multi-view tracking with silhouettes. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 716–721, 1999.

[DT96] F. W. DePiero and M. M. Trivedi. 3-d computer vision using structured light: Design, calibration, and implementation issues. *Advances in Computers*, 43:243–278, 1996.

[FARW99] R. B. Fisher, A. P. Ashbrook, C. Robertson, and N. Werghi. A low-cost range finder using a visually located, structured light source. In *Proceedings of the 2nd International Conference on 3-D Digital Imaging and Modeling*, pages 24–33, October 1999.

[Fau93] O. Faugeras. *Three-Dimensional Computer Vision*. M.I.T. Press, November 1993.

[HS91] R. M. Haralick and L. G. Shapiro. Glossary of computer vision terms. *Pattern Recognition*, 24(1):69–93, 1991.

[Imma] Immersion corporation. http://www.immersion.com.

[Immb] Lightscribe. http://www.immersion.com/products/3d/capture/lsinfo.shtml.

[JZ01] Q. Ji and Y. Zhang. Camera calibration with genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 31(2):120–130, March 2001.

[Kam99] M. Kampel. Tiefendatenregistrierung von rotationssymmetrischen Objekten. Master's thesis, Vienna University of Technology, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, Vienna, Austria, Februar 1999.

[KJV85] H. S. Kim, R. C. Jain, and R. A. Volz. Object recognition using multiple views. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 28–33, 1985.

[KK94] K. Kemmotsu and T. Kanade. Sensor placement design for object pose determination with three light-stripe range finders. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1357–1364, 1994.

[KS99a] M. Kampel and R. Sablatnig. On 3D modelling of archaeological sherds. In *Proceedings of International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging*, pages 95–98, 1999.

[KS99b] M. Kampel and R. Sablatnig. Range image registration of rotationally symmetric objects. In N. Brändle, editor, *Proceedings of the Computer Vision Winter Workshop*, pages 69–77, 1999.

[KS00] K. Kutulakos and S. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):197–216, July 2000.

[KSC01] M. Kampel, R. Sablatnig, and E. Costa. Classification of archaeological fragments using profile primitives. In *Computer Vision, Computer Graphics and Photogrammetry — a Common Viewpoint, Proceedings of the 25th Workshop of the Austrian Association for Pattern Recognition (ÖAGM)*, pages 151–158, 2001.

[Lis99] C. Liska. Das Adaptive Lichtschnittverfahren zur Oberflächenkonstruktion mittels Laserlicht. Master's thesis, Vienna University of Technology, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, Vienna, Austria, April 1999.

[LPC+00] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of SIGGRAPH*, pages 131–144, 2000.

[LS00] C. Liska and R. Sablatnig. Adaptive 3D acquisition using laser light. In T. Svoboda, editor, *Proceedings of Czech Pattern Recognition Workshop*, pages 111–116, 2000.

[MA83] W. N. Martin and J. K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[MB93] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):417–432, May 1993.

[MC96] E. Marchand and F. Chaumette. Controlled camera motions for scene reconstruction and exploration. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 169–176, June 1996.

[MN78] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. In *Proceedings of the Royal Society of London B*, number 200, pages 269–294, 1978.

[MR95] A. Marshal and D. Roberts. Automatically planning the inspection of three-dimensional objects using stereo computer vision. In *Proceedings of SPIE International Symposium on Intelligent Systems and Advanced Manufacturing*, 1995.

[Nal93] V. S. Nalwa. *A Guided Tour Of Computer Vision.* Addison-Wesley, 1993.

[NB86] V. S. Nalwa and T. O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:699–714, November 1986.

[Nie94] W. Niem. Robust and fast modelling of 3D natural objects from multiple views. In *Image and Video Processing II, Proceedings of SPIE*, pages 388–397, 1994.

[NSI99] K. Nishino, Y. Sato, and K. Ikeuchi. Appearance compression and synthesis based on 3D model for mixed reality. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 38–45, 1999.

[OTV93] C. Orton, P. Tyers, and A. Vince. *Pottery in Archaeology*, 1993.

[PDK01] J. Park, G. N. DeSouza, and A. C. Kak. Dual-beam structured-light scanning for 3-D object modeling. In *Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling*, pages 65–72, May 2001.

[Pit99] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1016–1030, October 1999.

[Pot87] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40:1–29, 1987.

[Rob96] L. Robert. Camera calibration without feature extraction. *Computer Vision and Image Understanding*, 63(2):314–325, March 1996.

[SA90] S. K. Srivastava and N. Ahuja. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics, and Image Processing*, 49:68–84, 1990.

[Shi87] Y. Shirai. *Three-Dimensional Computer Vision.* Springer-Verlag, 1987.

[SKS01] K. Schindler, M. Kampel, and R. Sablatnig. Fitting of a closed planar curve representing a profile of an archaeological fragment. In *Proceedings of International EuroConference on Virtual Reality, Archaeology and Cultural Heritage*, Athens, Greece, November 2001. in press.

[SM96] R. Sablatnig and C. Menard. Computer based acqusition of archaeological finds: The first step towards automatic classification. In *Proceedings of the 3rd International Symposium on Computing and Archaeology*, pages 429–446, 1996.

[Sze93] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993.

[TAT95] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[TG95a] G. Tarbox and S. Gottschlich. IVS: An integrated volumetric inspection system. *Computer Vision and Image Understanding*, 61:430–444, May 1995.

[TG95b] G. Tarbox and S. Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 61:84–111, January 1995.

[Tos00] S. Tosovic. Shape from silhouette. Technical Report PRIP-TR-064, Vienna University of Technology, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, Vienna, Austria, 2000.

[TS01] S. Tosovic and R. Sablatnig. 3D modeling of archaeological vessels using shape from silhouette. In *Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling*, pages 51–58, May 2001.

[Tsa86] R. Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[TTK91] K. A. Tarabanis, R. Y. Tsai, and A. Kaul. Computing viewpoints that satisfy optical constraints. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–158, 1991.

[TWVC99] M. Takatsuka, G. A. W. West, S. Venkatesh, and T. M. Caelli. Low-cost interactive active monocular range finder. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages I:444–449, 1999.

[VA86] J. Veenstra and N. Ahuja. Efficient octree generation from silhouettes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 537–542, 1986.

[Wat99] A. Watt. *3D Computer Graphics*. Addison Wesley, 3rd edition, December 1999.

[WC01] K. Y. K. Wong and R. Cipolla. Structure and motion from silhouettes. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, pages 217–222, 2001.

[WF94] P. Whaite and F. P. Ferrie. Autonomous exploration: Driven by uncertainty. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 339–346, 1994.

[Wil] R. G. Willson. Tsai camera calibration software. http://www.cs.cmu.edu/~rgw/TsaiCode.html.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.