TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

# Robot-Assisted Reconstruction of Objects Based on Iterative Geometric Approximation

## DIPLOMA THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Univ.-Prof. Dr. techn. A. Kugi
N. Đukić, MSc.

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by
Martin Fraunhofer
Matriculation number 01608051

Vienna, January 2022

**Complex Dynamical Systems Group**
A-1040 Wien, Gußhausstr. 27–29, Internet: https://www.acin.tuwien.ac.at

# Preamble

I feel thankful for the valuable insights I gained during the courses at the Automation and Control Institute of Vienna University of Technology. My interest in robotics was influenced by Univ.-Prof. Dr. techn. Andreas Kugi, who inspired me to write my thesis in this exciting field.

I want to express my deepest gratitude to my supervisor Nikola Đukić, MSc. for the helpful discussions during our meetings. Especially the motivational words encouraged me to work harder. I sincerely thank Dipl.-Ing. Dr. techn. Christian Hartl-Nesic for his commitment and interest in my work. Moreover, I appreciate the effort of Dipl.-Ing. Christoph Unger, who assisted me with the robot.

Throughout my education, my family supported me with many words of encouragement, for which I owe special thanks. In particular, I am indebted to Tanja for her enduring patience. Finally, I am very grateful to my parents, who spare no effort to support me in my plans.

Vienna, January 2022

# Abstract

In this work, a robotic 3D reconstruction system is developed for an experimental setup at TU Wien. This setup consists of a 3D camera, a seven-axis collaborative robot and two linear axes.

On the one hand, the developed reconstruction system can scan objects from manually predefined views. An automatic sequence optimization is presented to minimize the duration of the reconstruction process. Furthermore, an autonomous mode of operation is developed. For this purpose, a novel view planning algorithm is presented that determines suitable views based on an iteratively refined geometric approximation of the object to be scanned. Concepts of sequence optimization are revisited to select the next best view. The view planning algorithm eliminates the need to specify views manually, enabling a reconstruction of unknown objects with minimal human interaction.

Simulation studies demonstrate the suitability of the implemented algorithms. In addition, the developed reconstruction system is tested on an existing robotic system without linear axes to validate this work in experiments. It is shown that the proposed view planning algorithm is capable of scanning unknown objects with high accuracy.

# Kurzzusammenfassung

In dieser Arbeit wird ein robotergestütztes 3D Rekonstruktionssystem für einen experimentellen Aufbau an der TU Wien entwickelt. Dieser Aufbau besteht aus einer Tiefenkamera, einem siebenachsigen kollaborativen Roboter und zwei Linearachsen.

Einerseits können mit dem entwickelten Rekonstruktionssystem Objekte von zuvor manuell definierten Kameraposen gescannt werden. Hierfür wird eine automatische Reihenfolgeoptimierung vorgestellt, um die Dauer des Rekonstruktionsprozesses zu minimieren. Des Weiteren wird ein autonomer Betriebsmodus entwickelt. Dazu wird ein neuartiger View Planning-Algorithmus präsentiert, der geeignete Kameraposen anhand einer iterativ verbesserten geometrischen Approximation des zu scannenden Objekts wählt. Konzepte der Reihenfolgeoptimierung werden wieder aufgegriffen, um die nächstbeste Kamerapose zu selektieren. Der View Planning-Algorithmus eliminiert die Notwendigkeit, Kameraposen manuell festzulegen und ermöglicht somit eine Rekonstruktion unbekannter Objekte mit minimaler menschlicher Interaktion.

Simulationsstudien demonstrieren die Eignung der implementierten Algorithmen. Darüber hinaus wird das entwickelte Rekonstruktionssystem an einem bestehenden robotischen System ohne Linearachsen getestet, um diese Arbeit in Experimenten zu validieren. Es zeigt sich, dass der vorgeschlagene View Planning-Algorithmus in der Lage ist, unbekannte Objekte mit hoher Genauigkeit zu erfassen.

# Contents

# 1 Introduction

Digital three-dimensional (3D) models of physical objects are of interest in various fields such as medicine, archaeology, architecture, or manufacturing [1–4]. Developments over the last decade led to affordable 3D sensing devices [5] and paved the path for new applications, for instance in fashion or augmented reality [6, 7]. It is desirable to create 3D models in a time-efficient and reproducible manner, which motivates the use of industrial robots. In this thesis, a robot-assisted reconstruction system is developed for a new robotic system at TU Wien.

Figure 1.1 depicts a visualization of the planned robotic system which forms the basis of this work. It is composed of the seven-axis collaborative robot KUKA LBR iiwa 14 R820 equipped with the depth camera PHOTONEO MotionCam-3D M. The robot manipulator is mounted headlong on two linear axes, which enable the movement of the robot's base within the workcell. Thus, the total number of Degrees of Freedom (DoF) of this mechanical setup is nine. This configuration increases the available workspace of the robot and offers the possibility to move the camera around the object to be scanned.

## 1.1 Related Work

Automated reconstruction systems proposed in the literature essentially differ in the way how views for the 3D sensing device are planned and realized. A simple approach is to keep the sensor static and rotate the object of interest with a turntable [8]. In [9], an additional DoF in form of a cylindrical scanner is considered. Objects of complex shape cannot be fully reconstructed with such mechanical configurations. Thus, the scanning device was mounted on industrial robots in later works.

The ROBOSCAN system presented in [10] uses the robot ABB IRB 4400 with six axes in combination with a turntable. The main motivation for this work was to mitigate three bottlenecks in human-assisted 3D reconstruction, namely the selection of views, the positioning of the scanner, and the alignment of the captured scans. The first bottleneck is tackled by a view planning algorithm that automatically detects unsampled regions after an initial coverage phase. For this initial coverage, only a bounding volume enclosing the object is required. The latter bottleneck, i. e. the alignment of scans, is eliminated because the robot is calibrated and thus the pose of the scanner can be accurately determined by the measured joint angles. A disadvantage of ROBOSCAN is that motions for the robot are planned on dedicated hardware. Hence, the system cannot be easily extended by linear axes or otherwise adapted.

The aim of [11] was to develop an automated system for the reconstruction of cultural heritage. A system composed of the industrial robot FANUC LRMate 200i with six DoF, a linear axis, and a turntable is presented. With the linear axis, it is possible to lift the robot to scan objects of a maximum height of 2.5 m. The work was influenced by

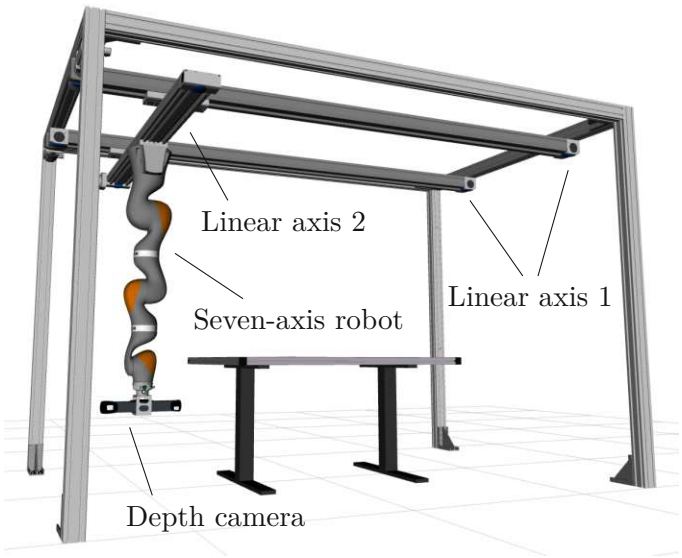



Figure 1.1: Robotic system composed of a collaborative robot mounted on two linear axes. The end-effector is a depth camera to scan physical objects on the table.

RoboScan, but a custom algorithm for planning motions was implemented to overcome the limitations on the extensibility of RoboScan. However, this algorithm is very specific and may not find a feasible path for more complicated environments. Especially, this is problematic for this thesis because no turntable is used and thus the cylindrical setup like in [11] is not available. Also, the view planning algorithm is tailored to the application of cultural heritage preservation where the scanning range of the sensing device is typically much smaller than the object to be scanned.

In [12, 13], a system to reconstruct objects for reverse engineering is considered. A hardware setup similar to RoboScan is used, but the robot does not have to stop for scanning. The novelty is the ability to scan along curved paths. Another system with a 6-DoF robot in combination with a turntable is found in [14]. It tackles the problem of view planning by classifying the scans into *well visible* and *barely visible* and then calculating the next best view based on the barely visible views. More recently, [15] proposed a robotic system for the application of inspection – also using a 6-DoF robot and a turntable. This publication presents a procedure consisting of the three phases *exploration*, *scan* and *rescan*. The objective is to perform the inspection in a time-efficient way by starting with a rough model for view planning and to increase the quality at a later stage if necessary. Paths for the robot are planned on a hemisphere centered at the turntable.

All these works have the objective of an automated reconstruction process in common. Their primary focus is on the selection of good views for the 3D sensing device. However, a reconstruction system consists of several additional required components, e. g. for planning motions and aligning scans. The development of this foundation for the given robotic system is the aim of this work, which is made more precise in the following.

## 1.2 Aim of this Work

The biggest difference between the work presented in this thesis and the works mentioned in Section 1.1 is the mechanical setup. In this work, the total DoF is nine, whereas most of the publications discussed previously consider robotic systems with seven DoF. Also, no turntable is used to rotate the object, i. e. the robot has to move around the object to perform a full scan. This imposes additional challenges for the positioning of the camera. Thus, the problem of planning collision-free paths for the robot is solved first.

Next, a reconstruction pipeline is to be developed which takes care of the alignment of scans and reconstructs the surface of the object. To start with, a list of desired camera views is assumed as given. The functionality is extended by an optional sequence optimization, which tries to shorten the duration of the scanning process. These concepts may be applied in a more general context, e. g. for the inspection of an object.

A further objective of this work is to provide a means to perform an object reconstruction with minimal human interaction. To this end, a view planning algorithm for scanning unknown objects is proposed. This eliminates the need to set views manually. It is based on an ellipsoidal approximation of the object. The approximation is obtained from an initial scan and then refined iteratively during reconstruction. Concepts of sequence optimization are revisited to select the next best view. Only a bounding volume of the object is assumed to be known to avoid collisions.

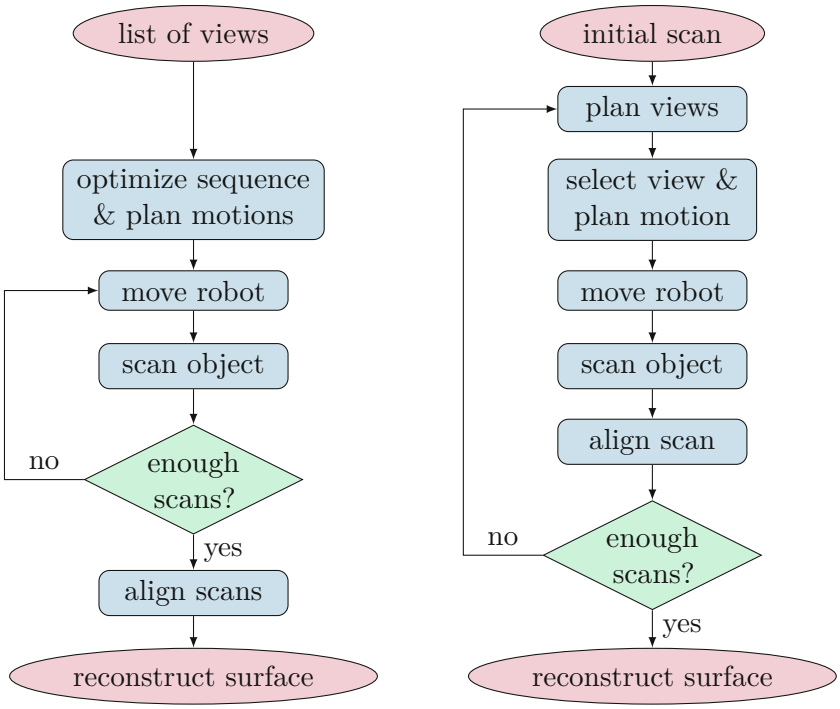
Finally, the operability of the developed reconstruction system is to be demonstrated in an experimental setup. For this purpose, the system is implemented utilizing the existing infrastructure of the robotic system. Prototypical scenarios are considered for validation of the proposed reconstruction system.

## 1.3 Thesis Outline

Chapter 2 introduces key concepts and methods applied in this thesis. The first part is concerned with view planning, followed by a discussion of motion planning. Further aspects of moving the camera by the robot are considered before an explanation of the actual object reconstruction process is given. Next, Chapter 3 describes the deployed software libraries and gives an overview of the system's implementation. Subsequently, Chapter 4 presents simulation studies that demonstrate the applicability of the selected algorithms. In Chapter 5, the developed reconstruction system is validated in an experimental setup. Finally, Chapter 6 concludes this thesis with a summary and a discussion of the obtained results. An outlook on possible future work is given.

# 2 Concept

This chapter presents the key elements of the developed reconstruction system from a conceptual perspective. The sections are organized in an order that tries to follow the steps involved in the process of creating a 3D model of a physical object. To get an overview, Figure 2.1 shows flow charts of two possible modes of operation.




(a) Model-based mode. (b) Non-model-based mode.

Figure 2.1: Two modes of operation for the developed reconstruction system: (a) Views are specified in advance, (b) Views are planned during execution.

For the first variant, depicted in Figure 2.1a, views for the camera are specified in advance. This is applicable for scanning objects where an initial 3D model is available. Correspondingly, this mode of operation is called the *model-based mode*. Because the views are determined before execution, it is possible to optimize the sequence in which the scans are made. After the computation of appropriate motions, the robot executes a part of the planned path and the camera scans the object. These steps repeat until enough scans have been made. Finally, the scans are aligned and the surface of the scanned object is reconstructed.

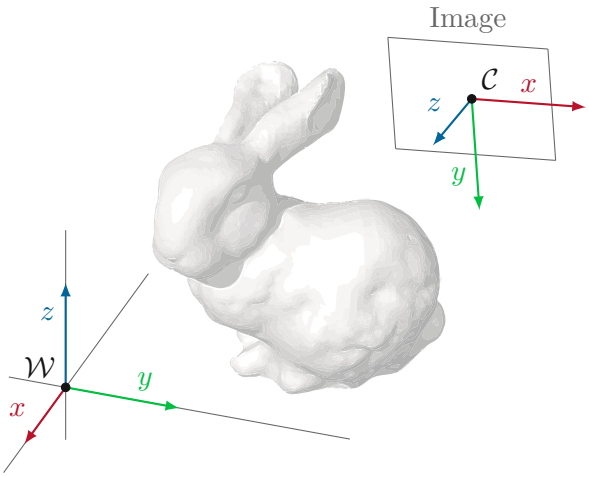



Figure 2.2: Definition of world frame $\mathcal{W}$ and camera frame $\mathcal{C}$.

To scan unknown objects without an initial 3D model, an alternative mode of operation is developed, called the *non-model-based mode*. The corresponding flow chart is illustrated in Figure 2.1b. Here, the reconstruction process starts with an initial scan. Views are planned and selected iteratively to react to new information gained during the scanning process. The remaining processing steps of the non-model-based mode are analogous to the model-based variant, except that scans are aligned in each iteration for planning views in the next iteration.

## 2.1 View Planning

Due to the limited field of view of a depth camera and self-occlusions of the physical object, it is necessary to make multiple scans from distinct views. The process of determining this set of views is called *view planning*. In the literature, approaches to view planning have been proposed which try to choose the views in a somehow optimal way. They can be classified in whether or not an initial 3D model of the object exists [16]. Our proposed method addresses both of these cases. Examples of model-based and non-model-based view planning are [17] and [18], respectively. This distinction is reflected in the model-based and non-model-based mode introduced previously. Therefore, these modes are discussed separately after presenting basic coordinate frames.

### 2.1.1 Coordinate Frames

To express the camera's view to the object of interest, two coordinate frames, the *world frame* $\mathcal{W}$ and the *camera frame* $\mathcal{C}$, are introduced. Whereas the object may be placed arbitrarily in the world $\mathcal{W}$, the camera is usually fixed to the camera frame $\mathcal{C}$ as illustrated in Figure 2.2. The image sensor coincides with the origin of $\mathcal{C}$ and the direction of view is aligned with the $z$-axis. Relative to the captured image, the camera frame's $x$- and $y$-axis point to the right and bottom, respectively.

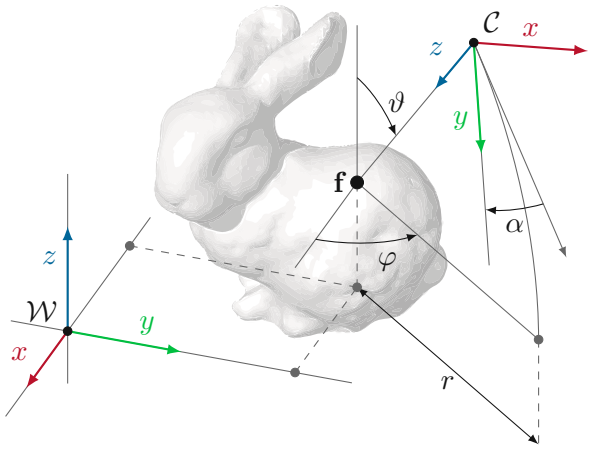



Figure 2.3: Geometric relation between frames $\mathcal{W}$ and $\mathcal{C}$. The camera's pose relative to $\mathcal{W}$ is determined by $\mathbf{f}$ and $r$ as well as the angles $\varphi$, $\vartheta$, and $\alpha$.

The geometric relation between the frames $\mathcal{W}$ and $\mathcal{C}$ is expressed by means of a homogeneous transformation $\mathbf{H}^{\mathcal{C}}_{\mathcal{W}}$. In general, a homogeneous transformation $\mathbf{H}^{\mathcal{Y}}_{\mathcal{X}}$ describes the position and orientation of the coordinate frame $\mathcal{Y}$ in the coordinates of $\mathcal{X}$ and can be written as

$$\mathbf{H}^{\mathcal{Y}}_{\mathcal{X}} = \begin{bmatrix} \mathbf{R}^{\mathcal{Y}}_{\mathcal{X}} & \mathbf{d}^{\mathcal{Y}}_{\mathcal{X}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.1}$$

In this expression, the orthogonal rotation matrix $\mathbf{R}^{\mathcal{Y}}_{\mathcal{X}} \in \mathrm{SO}(3)$ represents the orientation of $\mathcal{Y}$ and $\mathbf{d}^{\mathcal{Y}}_{\mathcal{X}} \in \mathbb{R}^3$ denotes the position of its origin. The last row of $\mathbf{H}^{\mathcal{Y}}_{\mathcal{X}}$ is such that homogeneous transformations can be applied successively with intermediate frames, i. e.

$$\mathbf{H}^{\mathcal{Z}}_{\mathcal{X}} = \mathbf{H}^{\mathcal{Y}}_{\mathcal{X}} \mathbf{H}^{\mathcal{Z}}_{\mathcal{Y}}. \tag{2.2}$$

### 2.1.2 Model-Based Mode

The actual planning of views based on an initial 3D model is not treated in this thesis. Rather, the interface to the model-based mode of the developed reconstruction system is a list of desired views for the camera. This list may be specified by hand. To this end, the following geometric representation of views is adopted, see Figure 2.3. The camera's position is given in sphere coordinates $(r, \vartheta, \varphi)$ relative to a focus point $\mathbf{f} \in \mathbb{R}^3$. Because the camera points toward the focus point $\mathbf{f}$, its orientation is already partially defined. The remaining degree of freedom is the rotation of the camera around its $z$-axis, given by the angle $\alpha$. By varying the angles $\varphi$, $\vartheta$, and $\alpha$ while keeping $\mathbf{f}$ and $r$ fixed, views around an object can be specified intuitively. This is the reason for utilizing this representation also for the user interface of the reconstruction system, see Section 3.3.

### 2.1.3 Non-Model-Based Mode

The non-model-based mode plans views during the scanning process. In this work, a novel view planning algorithm is proposed, which is based on an ellipsoidal approximation of

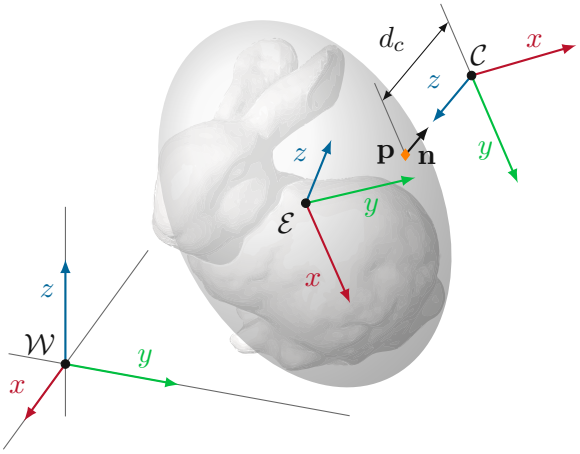



Figure 2.4: Estimated ellipsoid with frame $\mathcal{E}$. A point $\mathbf{p}$ on the surface determines the pose of the camera frame $\mathcal{C}$.

the object to be scanned. This ellipsoid is estimated iteratively starting with an initial scan. After each subsequent scan, the estimate is improved as more information about the object is gained. The algorithm terminates when a desired number of $N$ scans has been made. Only a bounding volume of the object is assumed to be known to avoid collisions. The following text describes the algorithm in more detail.

**Ellipsoid.** The starting point of the view planning algorithm is an initial scan of the object. An ellipsoid is computed that encompasses the observed part of the object. For the following explanation, a new coordinate frame, called the *ellipsoid frame* $\mathcal{E}$, is introduced. This frame $\mathcal{E}$, illustrated in Figure 2.4, is located at the center of the ellipsoid and aligned with its principal axes. The surface of the ellipsoid is then given by the expression

$$f(x, y, z) = \frac{x^2}{e_1^2} + \frac{y^2}{e_2^2} + \frac{z^2}{e_3^2} - 1 = 0, \tag{2.3}$$

where $e_1$, $e_2$, and $e_3$ denote the extent of the ellipsoid in $x$-, $y$-, and $z$-direction, respectively.

**From Surface to Pose.** Given a point $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$ on the surface of the ellipsoid and the corresponding normal

$$\mathbf{n} = \nabla f = 2\begin{bmatrix} \frac{x}{e_1^2} & \frac{y}{e_2^2} & \frac{z}{e_3^2} \end{bmatrix}^{\mathrm{T}}, \tag{2.4}$$

the position of the camera relative to $\mathcal{E}$ is set to

$$\mathbf{d}_{\mathcal{E}}^{\mathcal{C}} = \mathbf{p} + d_c \frac{\mathbf{n}}{\|\mathbf{n}\|}. \tag{2.5}$$

Here, the parameter $d_c > 0$ denotes the desired distance of the camera to the ellipsoid, see Figure 2.4. In addition to the position $\mathbf{d}_{\mathcal{E}}^{\mathcal{C}}$, the orientation of the camera is defined. The

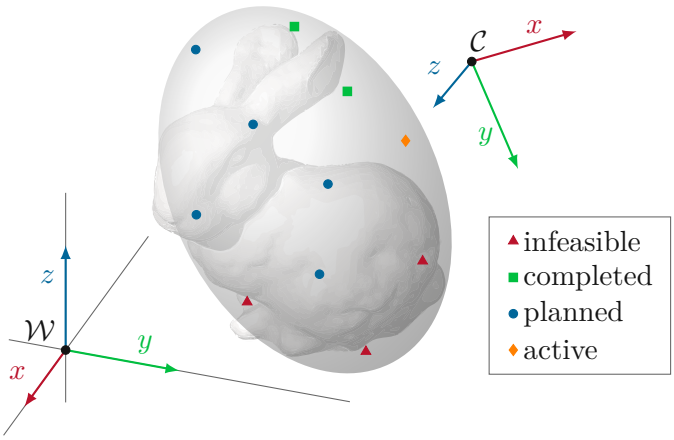



Figure 2.5: Views distributed on the ellipsoid. They are classified into infeasible, completed, and planned views. The active view is selected for the next scan.

rotation matrix $\mathbf{R}_{\mathcal{E}}^{\mathcal{C}}$ is chosen such that the $z$-axis of $\mathcal{C}$ points in the negative direction of $\mathbf{n}$, i. e. the camera is directed to the object, and the $x$-axis of $\mathcal{C}$ is perpendicular to the $z$-axis of $\mathcal{W}$, i. e. the camera is positioned horizontally.

**Sampling the Surface.** To determine a suitable point $\mathbf{p}$ on the ellipsoid, the surface is sampled, depicted in Figure 2.5. Let $\mathcal{A}$ denote the set of all previous valid samples. A valid sample $\mathbf{p}$ is computed by first randomly generating a set $\mathcal{B}$ of $M$ preliminary samples. For each of these preliminary samples $\mathbf{s}$, the cost function

$$c(\mathbf{s}) = \sum_{\mathbf{p}' \in \mathcal{A}} \frac{1}{\|\mathbf{s} - \mathbf{p}'\|_{\mathcal{E}}^2 + \sigma} \tag{2.6}$$

with regularization term $\sigma > 0$ and norm $\|\cdot\|_{\mathcal{E}}$ (defined below) is evaluated. This cost function $c(\mathbf{s})$ penalizes preliminary samples in the vicinity to previous valid samples and can be interpreted as an artificial potential field ensuring a good distribution of views. The preliminary sample $\mathbf{s}$ with smallest $c(\mathbf{s})$ is then selected as next valid sample,

$$\mathbf{p} = \arg\min_{\mathbf{s} \in \mathcal{B}} \{c(\mathbf{s})\}. \tag{2.7}$$

The norm $\|\cdot\|_{\mathcal{E}}$ is defined as

$$\|\mathbf{p}\|_{\mathcal{E}} = \sqrt{\frac{x^2}{e_1^2} + \frac{y^2}{e_2^2} + \frac{z^2}{e_3^2}}. \tag{2.8}$$

This is equivalent to transforming the ellipsoid to a sphere and then calculating the Euclidean distance of points. Thus, the resulting cost is independent of the ratio of the quantities $e_1$, $e_2$, and $e_3$.

**Planned Poses.** For each valid sample $\mathbf{p}$, the corresponding camera pose $\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}(\mathbf{p})$ is determined as described previously. Depending on whether the pose $\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}(\mathbf{p})$ is feasible for the robot or not, it is inserted in the set $\mathcal{H}_p$ of the *planned poses* or in the set $\mathcal{H}_i$ of the *infeasible poses*. In any case, the corresponding sample $\mathbf{p}$ is added to the set $\mathcal{A}$ of all valid samples. When $N - |\mathcal{H}_c|$ feasible poses have been found, where $|\mathcal{H}_c|$ denotes the number of completed scans, the algorithm proceeds as follows.

**Selecting a Pose.** A pose within the set $\mathcal{H}_p$ of planned poses is chosen for the next scan. This is done by selecting the pose that is "closest" to the current robot configuration to ensure a fast scanning process. More details on this aspect are given in Section 2.3.3. Thereafter, the object is scanned and a new ellipsoid is estimated. The selected pose, also called the *active pose*, is then added to the set $\mathcal{H}_c$ of the *completed poses*.

**From Pose to Surface.** The estimate of the ellipsoid changes every iteration as new information about the object becomes available. Therefore, the completed and infeasible poses do not fit the new ellipsoid anymore. This is relevant for the set $\mathcal{A}$, which governs the distribution of the following samples. The discrepancy is resolved by calculating the points on the new ellipsoid's surface that are closest to the positions of the poses in $\mathcal{H}_c \cup \mathcal{H}_i$. For this purpose, the algorithm presented in [19] is applied. The computed nearest points constitute the new set $\mathcal{A}$ for the next iteration.

Table 2.1 summarizes the described view planning algorithm.

## 2.2 Motion Planning

The goal of *motion planning* is to find a collision-free path for the robot to fulfil its task. In the context of this thesis, a fundamental task for the robot is to move the camera to desired poses in its workspace. Such high-level goal specifications serve as input to planning algorithms, which compute the needed movements for each individual joint. Robots with high DoF, like the system considered in this work, are particularly challenging for motion planning. In addition, obstacles in the robot's workspace further complicate the problem to be solved.

In the following, the motion planning problem is introduced. Subsequently, fundamental concepts of motion planning are explained and an overview of existing approaches is given. After selecting a suitable class of motion planning algorithms, an appropriate mathematical model of the robot is presented and aspects of collision checking are mentioned. Finally, the execution of time-parameterized paths is discussed.

### 2.2.1 Robot Configurations

The joint positions $q_i$, $i = 1, \ldots, n$ of a robot with $n$ DoF are summarized in the vector

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^{\mathrm{T}}, \tag{2.9}$$

which is also called the *robot configuration*. In its basic formulation, the goal of motion planning is to find a collision-free path from a start configuration $\mathbf{q}_s$ to a goal configura-

$\mathcal{H}_c \leftarrow \{\}$
$\mathcal{H}_i \leftarrow \{\}$
make initial scan
estimate ellipsoid
**while** $|\mathcal{H}_c| < N$
  $\mathcal{A} \leftarrow \{\}$
  **for** every $\mathbf{H}_{\mathcal{E}}^{\mathcal{C}} \in \mathcal{H}_c \cup \mathcal{H}_i$
    find $\mathbf{p}$ on the ellipsoid's surface that is closest to $\mathbf{d}_{\mathcal{E}}^{\mathcal{C}}$ of $\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{p}\}$
  $\mathcal{H}_p \leftarrow \{\}$
  **while** $|\mathcal{H}_p| < N - |\mathcal{H}_c|$
    $\mathcal{B} \leftarrow$ sample $M$ points on ellipsoid's surface
    **for** every $\mathbf{s} \in \mathcal{B}$
      calculate cost of $\mathbf{s}$ based on $\mathcal{A}$
    $\mathbf{p} \leftarrow \mathbf{s} \in \mathcal{B}$ with smallest cost $c(\mathbf{s})$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{p}\}$
    **if** $\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}(\mathbf{p})$ is feasible for the robot
      $\mathcal{H}_p \leftarrow \mathcal{H}_p \cup \{\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}\}$
    **else**
      $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup \{\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}\}$
  select a $\mathbf{H}_{\mathcal{E}}^{\mathcal{C}} \in \mathcal{H}_p$ (see Section 2.3.3)
  move camera and make scan
  estimate new ellipsoid
  $\mathcal{H}_c \leftarrow \mathcal{H}_c \cup \{\mathbf{H}_{\mathcal{E}}^{\mathcal{C}}\}$

Table 2.1: Procedure of the view planning algorithm for the non-model-based mode.

tion $\mathbf{q}_g$. For this thesis, the concrete goal configuration $\mathbf{q}_g$ is of minor importance. Rather, a pose goal for the camera, i. e. a desired transformation $\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}$ is of interest. The considered robotic system has nine DoF and thus exhibits a redundancy in the 3D Cartesian space of dimension six. Due to this redundancy, there is generally an infinite number of goal configurations $\mathbf{q}_g$ that lead to a given transformation $\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}$. Finding configurations $\mathbf{q}$ for a prescribed $\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}(\mathbf{q})$ is referred to as *inverse kinematics problem*. Essentially, the goal configuration $\mathbf{q}_g$ for motion planning can be set to any solution of the inverse kinematics problem.

### 2.2.2 Planning Algorithms

A fundamental concept in motion planning is the notion of the configuration space [20], denoted by $\mathcal{X}$. It can be considered as a special state space that comprises the set of all possible robot configurations $\mathbf{q}$. Due to obstacles in the robot's workspace and possible self-collision, a subset $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$, called the *obstacle region*, corresponds to invalid robot configurations. The other part $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ of $\mathcal{X}$, which contains the valid configurations, is called the *free space*.

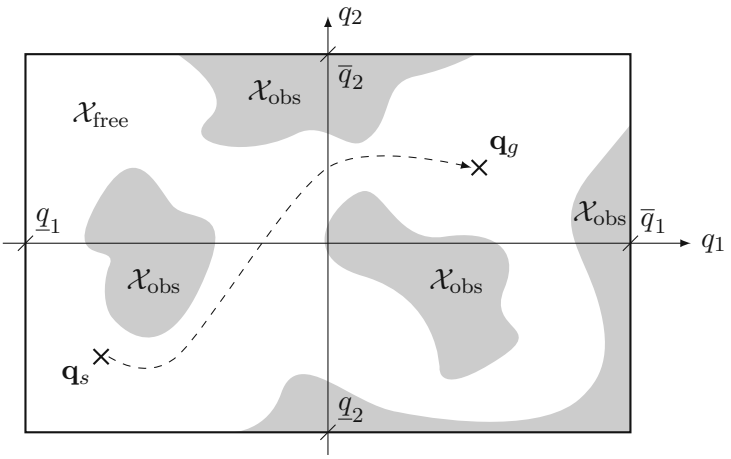



Figure 2.6: Illustration of a configuration space $\mathcal{X}$ for a manipulator with two joints. A possible path from start configuration $\mathbf{q}_s$ to goal configuration $\mathbf{q}_g$ is indicated by a dashed line.

Figure 2.6 illustrates this concept for a two-dimensional configuration space $\mathcal{X}$, related to an imaginary manipulator with two joints $q_1$ and $q_2$. The set $\mathcal{X}$ of all possible configurations $\mathbf{q}$ is defined by the lower and upper joint limits, which restrict $q_1$ and $q_2$ to $\underline{q}_1 \leq q_1 \leq \overline{q}_1$ and $\underline{q}_2 \leq q_2 \leq \overline{q}_2$, respectively.

With the notion of the configuration space $\mathcal{X}$, the motion planning problem can be formulated as finding a path in the free configuration space $\mathcal{X}_{\text{free}}$ from start configuration $\mathbf{q}_s$ to goal configuration $\mathbf{q}_g$ [21]. Except for simple cases, the free space $\mathcal{X}_{\text{free}}$ has a complex shape, which is hard to calculate explicitly [22]. This is why many planning algorithms resort to implicit representations.

Early algorithms relying on implicit representations of the free space $\mathcal{X}_{\text{free}}$ made use of artificial potential fields [23]. These approaches are suitable for online planning, where obstacles are not known beforehand. However, the robot may get stuck in a local minimum of the potential field. Also, in this context, geometric information of the robot's environment is available, which is why offline planners can be used.

A central idea is to utilize a discretization of the configuration space [24]. Algorithms using this concept can be classified in *combinatorial* and *sampling-based* approaches [21]. Examples of combinatorial algorithms are planning via cell decomposition [25] and planning via retraction [26, 27]. Combinatorial algorithms provide the strong guarantee of *completeness*, meaning that they succeed if a solution exists and report failure otherwise in finite time. However, they are often impractical due to their computational complexity.

Sampling-based algorithms [21, 28] take samples in the configuration space $\mathcal{X}$ and evaluate whether the corresponding configurations are in a collision or not. A search in the configuration space is performed to find a sequence of samples that connects the start configuration $\mathbf{q}_s$ with the goal configuration $\mathbf{q}_g$. There are different strategies for how samples are selected. Often, probabilistic sampling techniques are used because they have shown good performance.

An advantage of sampling-based algorithms may be the fact that collision checking

is treated as a black box, i. e. it can be separated from the actual planning algorithm. Thus, the specific representation of obstacles is not that critical as for combinatorial planning algorithms. However, the strong guarantee of completeness needs to be relaxed for sampling-based algorithms. Randomized sampling-based algorithms are called *probabilistic complete*. This means that if a solution exists, the probability of success converges to one with sufficiently long runtime. Apart from the weaker guarantee on finding a solution, the class of sampling-based algorithms is highly efficient and well suited for high-dimensional problems. Therefore, sampling-based algorithms will be applied in this thesis.

It is worth mentioning that sampling-based motion planners often output a path that needs to be smoothed in a post-processing step. More recent approaches to motion planning incorporate dynamics and try to directly compute good paths or trajectories by means of optimization-based techniques. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [29], for example, uses an objective functional composed of two parts. One part penalizes proximity to obstacles whereas the complementary part penalizes undesired dynamical properties of the trajectory. The method proposed in [30] is similar to CHOMP, but uses another optimization method and evaluates collisions differently.

### 2.2.3 Kinematic Robot Model

The mathematical model presented here, which is used by the sampling-based planning algorithms, is purely kinematic. Dynamics are considered at a later stage, when the time-parameterized motion plan is executed, to be discussed in Section 2.2.5.

Figure 2.7 shows a visualization of the KUKA LBR iiwa 14 R820 mounted on the two linear axes, along with the coordinate frames $\mathcal{L}_i$, $i = 1, \dots, 9$ of the serial chain of links. Starting at the top, the first two joint positions $q_1$ and $q_2$ correspond to the prismatic joints of the linear axes. The following joint positions $q_3, \dots, q_9$ are related to the seven revolute joints of the KUKA LBR iiwa 14 R820, whose axes coincide with the $z$-axes of the frames $\mathcal{L}_3, \dots, \mathcal{L}_9$. At the end of the serial chain, the camera is mounted on the robot's flange, expressed by the fixed transformation $\mathbf{H}^{\mathcal{C}}_{\mathcal{L}_9}$. By applying transformations successively, the pose of each link can be expressed in the inertial world frame $\mathcal{W}$. For example, the transformation for the camera frame $\mathcal{C}$ is obtained as

$$\mathbf{H}^{\mathcal{C}}_{\mathcal{W}}(\mathbf{q}) = \mathbf{H}^{\mathcal{L}_1}_{\mathcal{W}}(q_1)\mathbf{H}^{\mathcal{L}_2}_{\mathcal{L}_1}(q_2)\cdots\mathbf{H}^{\mathcal{L}_9}_{\mathcal{L}_8}(q_9)\mathbf{H}^{\mathcal{C}}_{\mathcal{L}_9}. \tag{2.10}$$

### 2.2.4 Collision Checking

For collision checking, the kinematic model of the robot is augmented with geometric properties of the links. Since sampling-based algorithms possibly need to evaluate the collision state for a large number of samples, the computation time of a single collision check has a great impact on the overall performance of motion planning. To facilitate fast collision checks, it is desirable to model the links with simple objects, also called *primitive shapes*. However, a compromise needs to be made between a simple geometric shape and an accurate model that does not artificially restrict the freedom of movement. In this work, the links of the collaborative robot are modeled as *capsules*. A capsule is the union of two spheres with a cylinder in between. This is shown in Figure 2.8 for a link of the KUKA LBR iiwa 14 R820. By increasing the radius of the spheres, a safety

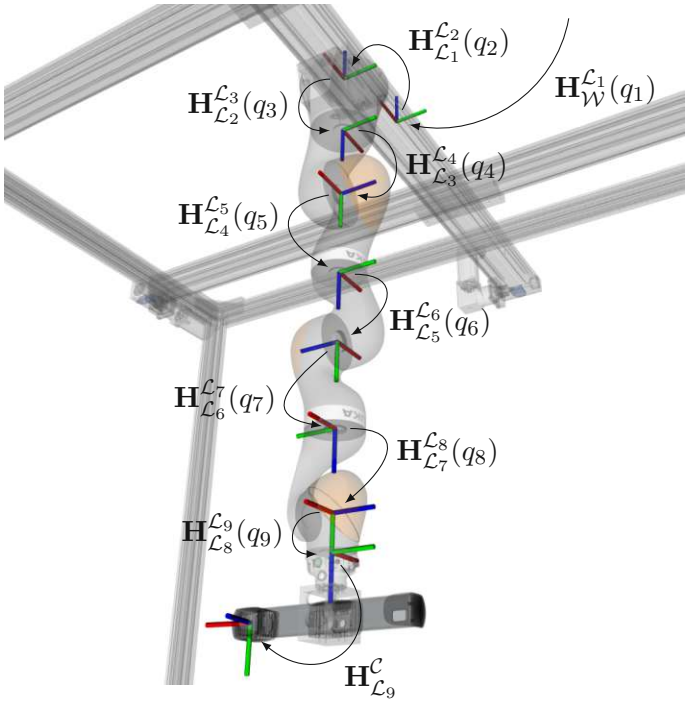



Figure 2.7: Drawing of the robotic system together with the coordinate frames of the links. Homogeneous transformations between successive links are illustrated with arrows.

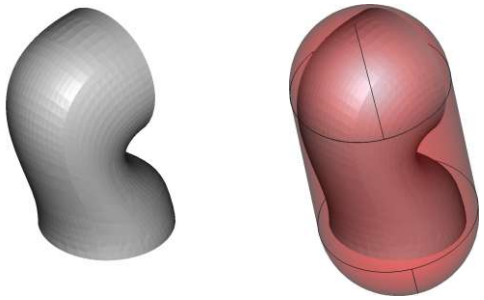
distance can be realized, i. e. that the robot keeps a minimum distance to obstacles. All other components of the physical setup, i. e. linear axes, table, camera, etc. are modeled as boxes. Again, a safety distance is realized by intentionally choosing the side length larger than the actual dimensions of the physical object.

### 2.2.5 Trajectory Execution

In this thesis, a trajectory tracking controller is deployed to let the robot execute the planned motions. For this purpose, the paths computed during motion planning are first parameterized in time. More details regarding this aspect are given in Chapter 3. This section briefly describes the robot model and the control law used to execute trajectories on the robotic system.

#### Dynamic Robot Model

For motion planning, a kinematic model of the robotic system augmented with geometric properties of the links suffices to apply sampling-based planning algorithms. However, to execute the planned trajectories, a model considering the dynamics of the system is needed. In this work, the existing dynamic model of the KUKA LBR iiwa 14 R820 is extended to incorporate the linear axes. The dynamic model is derived using the



(a) (b)

Figure 2.8: A link of the KUKA LBR iiwa 14 R820: (a) Actual geometric shape, (b) Capsule around the link used for collision checking.

Lagrangian formalism [31] and can be written in the form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{2.11}$$

with the positive definite mass matrix $\mathbf{M}(\mathbf{q})$, the CORIOLIS matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, the vector of gravitational forces $\mathbf{g}(\mathbf{q})$ and the generalized external forces $\boldsymbol{\tau}$.

#### Control Law

The control law is based on the computed torque [32]

$$\boldsymbol{\tau}_c = \mathbf{M}(\mathbf{q})\mathbf{v} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}), \tag{2.12}$$

where $\mathbf{v}$ denotes a new system input. With the error

$$\mathbf{e}_\mathbf{q} = \mathbf{q} - \mathbf{q}_d \tag{2.13}$$

between measured joint positions $\mathbf{q}$ and desired joint positions $\mathbf{q}_d$, the new input $\mathbf{v}$ is calculated according to

$$\mathbf{v} = \ddot{\mathbf{q}}_d - \mathbf{K}_0\mathbf{e}_\mathbf{q} - \mathbf{K}_1\dot{\mathbf{e}}_\mathbf{q} - \mathbf{K}_I \int_0^t \mathbf{e}_\mathbf{q}(\xi)\,\mathrm{d}\xi. \tag{2.14}$$

In this equation, $\mathbf{K}_0$, $\mathbf{K}_1$ and $\mathbf{K}_I$ are positive definite diagonal matrices. The stiffness and damping of the error dynamics can be adjusted with $\mathbf{K}_0$ and $\mathbf{K}_1$, respectively. The integral term with the matrix $\mathbf{K}_I$ accumulates the error $\mathbf{e}_\mathbf{q}$ over time to eliminate a static control deviation.

## 2.3 Sequence Optimization

The previous section introduced motion planning, which tries to find a feasible path for the robot to reach a desired camera pose. Generally, the task is to move the camera to *multiple* poses for scanning, which is the topic of this section. Here, a distinction between model-based and non-model-based mode is made because their starting point is different.

In the case of the model-based mode, the desired camera poses are specified in advance. A key observation is that individual scans are independent of each other, i. e. the order in which they are made does not influence the quality of the reconstruction. Therefore, the initial sequence may be reordered to optimize the system's performance in some sense. A survey on sequence optimization in the field of robotics is found, for example, in [33]. In this thesis, the problem of finding an optimal sequence is modeled as a Traveling Salesman Problem (TSP) [34].

For the non-model-based mode, the camera poses are determined during reconstruction, i. e. they are not known beforehand. Multiple camera poses are planned in each iteration and one of them is selected for the next scan. One possibility is to select the camera pose that is closest to the current robot configuration in some sense. This can be solved with the same concepts that are used for sequence optimization and is therefore also treated in this section.

In the following, the TSP is formulated in the context of this thesis. Then, the applied algorithm to approximately solve the TSP is briefly described. Next, the concepts of sequence optimization are specialized to the non-model-based mode. Finally, a possible distance metric is presented, which is used in this thesis.

### 2.3.1 Traveling Salesman Problem

Given a set of $m$ camera poses, the goal is to visit each pose exactly once while minimizing an appropriate additive distance metric $c$. Examples of possible distance metrics are the total time needed to complete the route or the energy consumption during the execution of the trajectory.

Figure 2.9 depicts an example in form of a weighted graph $G$ with $m = 4$ vertices. Vertex $\mathbf{q}_0$ visualizes the start configuration of the robot. The remaining $m$ vertices are inverse kinematics solutions $\mathbf{q}_i$, $i = 1, \dots, m$ corresponding to the desired $m$ camera poses. Each edge represents a path connecting a pair of robot configurations. Costs associated with the edges are denoted by $c_{ij}$ with $i = 0, \dots, m-1$ and $j = i+1, \dots, m$. In this simple example, the sequence which minimizes the total cost is $(\mathbf{q}_3, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_4)$, leading to a total cost of

$$c = c_{03} + c_{13} + c_{12} + c_{24} = 7. \tag{2.15}$$

Three challenges arise with this problem:

1. Generally, there are multiple solutions for the configurations $\mathbf{q}_i$ due to the redundancy of the robotic system. In this context, a particular set of solutions $\mathbf{q}_i$ must be selected before solving the TSP. This choice has a substantial influence on the weighted graph and thus on the solution of the TSP. Variants of the TSP exist where multiple solutions of the inverse kinematics problem can be incorporated into the problem formulation [35].

2. For general distance metrics, the cost associated with an edge depends on the specific course of the path, not only the start and the end. Therefore, the path needs to be planned before its cost can be computed. However, motion planning for high DoF is a time-consuming task and the number of edges grows with $\mathcal{O}(m^2)$. This leads to a considerable number of planning requests for a few dozens of scans.

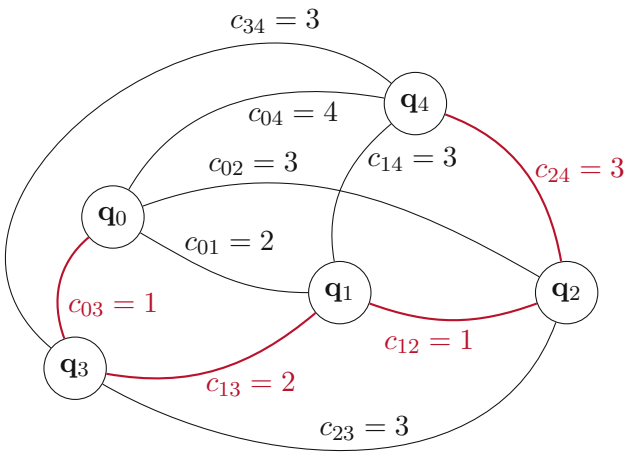



Figure 2.9: Example of the TSP represented as a weighted graph. The optimal route is highlighted.

3. If a specific set of inverse kinematics solutions $\mathbf{q}_i$ has been selected and costs $c_{ij}$ for the edges have been computed, it remains to actually solve the TSP. It is known that the TSP is NP-hard [34]. This motivates the use of algorithms that approximate the solution of the TSP.

### 2.3.2 Approximation Algorithm

The applied approximation algorithm [36] assumes that the time needed to compute a minimum spanning tree $T$ of the graph $G$ is negligible compared to planning a path. The procedure is summarized in Table 2.2 and explained in the following.

First, a complete graph $G$ is created with a set of vertices $\{\mathbf{q}_0, \ldots, \mathbf{q}_m\}$. Edges are inserted, but no motion plan is computed yet. Since the paths for the robot are therefore not known, the weights of the edges are initialized with lower bounds on the exact costs. Correspondingly, the edges are called *non-exact*. As an optional step, which is not part of the original algorithm [36], the graph $G$ may be optimized (discussed later on). Then, the main loop of the algorithm is executed. A minimum spanning tree $T$ is calculated and its cost is stored in the variable $\kappa$. The algorithm proceeds with an inner loop:

- As long as the cost of $T$ stays below $\beta\kappa$, where $\beta > 1$ is a parameter, the algorithm repeatedly picks a non-exact edge $e$ in $T$. A corresponding motion plan is computed and the cost of $e$ (and $T$) is updated. The edge $e$ is then called *exact*.
- If the cost of $T$ becomes greater than $\beta\kappa$, the computation of a minimum spanning tree is done again.

If all edges are exact, the algorithm performs a preorder traversal of $T$, i. e. it lists the vertices of $T$ in depth-first order. This is the final sequence and determines the order in

```
create complete graph G
for every edge e in G
    calculate lower bound on c(e)
    mark e as non-exact
optimize graph G (optional)
repeat
    T ← minimum spanning tree of G
    κ ← c(T)
    while c(T) ≤ βκ
        if all edges in T are exact
            P ← preorder traversal of T
            return paths corresponding to P
        else
            pick a non-exact edge e in T
            plan path corresponding to e
            update cost of e
            mark e as exact
```

Table 2.2: Procedure of the (modified) algorithm [36].

which the computed motion plans are returned. Under the assumption that the distance metric satisfies the triangle inequality, the cost of the computed sequence is at most $2\beta$ as large as the cost of the exact solution of the TSP. The parameter $\beta$ influences the number of planning requests. By increasing $\beta$, fewer planning requests are made because the condition $c(T) \leq \beta\kappa$ is more likely satisfied.

As mentioned in Section 2.3.1, the solution of the TSP greatly depends on the chosen set of robot configurations. To come up with a reasonably good choice of inverse kinematics (IK) solutions, the following optional step is performed ("optimize graph" in Table 2.2). The procedure is shown in Table 2.3 and starts with the graph $G$. It is tried to improve the graph $G$ by repeatedly iterating over the vertices and calculating new inverse kinematics (IK) solutions. If the new graph $G_{\text{new}}$ leads to a lower cost of the preorder traversal, the graph $G$ is replaced by $G_{\text{new}}$. Otherwise, the counter $k$ is incremented. If the heuristic to improve $G$ fails too often, the algorithm terminates and returns $G$. This behavior can be adjusted with the parameter $k_{\max} \in \mathbb{N}$.

### 2.3.3 View Selection

For the non-model-based mode, a single view needs to be selected out of the planned views in each iteration. One possibility is to select the view that is closest (in some sense) to the current robot configuration. This can be achieved with a specialization of the concepts discussed previously.

In particular, the algorithm in Table 2.4 is applied for view selection. It uses the same concept of exact and non-exact edges as the algorithm [36], but instead of a complete graph $G$, a tree $T$ is created. The tree $T$ has vertex $\mathbf{q}_0$ as its root and all other vertices

$T \leftarrow$ minimum spanning tree of $G$
$P \leftarrow$ preorder traversal of $T$
$k \leftarrow 0$
**repeat**
  **for** every vertex $\mathbf{q} \in \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$
    $G_{\text{new}} \leftarrow G$
    replace vertex $\mathbf{q}$ in $G_{\text{new}}$ by new IK solution
    update lower bounds of edges in $G_{\text{new}}$
    $T_{\text{new}} \leftarrow$ minimum spanning tree of $G_{\text{new}}$
    $P_{\text{new}} \leftarrow$ preorder traversal of $T_{\text{new}}$
    **if** $c(P_{\text{new}}) < c(P)$
      $G \leftarrow G_{\text{new}}$
      $P \leftarrow P_{\text{new}}$
      $k \leftarrow 0$
    **else**
      $k \leftarrow k + 1$
      **if** $k > k_{\max}$
        **return** $G$

Table 2.3: Procedure used to optimize $G$ for the algorithm in Table 2.2.

are leaves. This reflects the fact that only the next view needs to be selected instead of determining a route. Like the algorithm in Table 2.2, the edges are initialized with lower bounds on $c$ and marked as non-exact. Then, similarly to the optional step in Table 2.2, the tree $T$ may be optimized (discussed later on). Now, the task is to plan paths and look for the one that minimizes the distance metric. To this end, the main loop of the algorithm is executed. It terminates when an exact edge with the lowest cost has been determined.

Table 2.5 shows the algorithm for the optional optimization of $T$. Basically, the structure is the same as for the graph optimization algorithm in Table 2.3. The algorithm iteratively updates the inverse kinematics (IK) solution for each vertex in $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ and checks whether the total cost of the tree is reduced. Again, the parameter $k_{\max}$ determines the maximum number of unsuccessful trials.

### 2.3.4 Distance Metric

It is desirable to minimize the total duration of the scanning process. Therefore, the distance metric applied in this thesis is the duration $t_p$ of the planned trajectory. In the following, a lower bound on the duration $t_p$, denoted by $\underline{t}_p$, is derived under the assumption of acceleration limits $a_{\max,i}$ and velocity limits $v_{\max,i}$ for the robot's joints $i = 1, \ldots, n$. The lower bound $\underline{t}_p$ is needed by the algorithms described in Section 2.3.2 and Section 2.3.3 to initialize the weights of non-exact edges.

```
create tree T with root q_0 and leaves {q_1, ..., q_m}
for every edge e in T
    calculate lower bound on c(e)
    mark e as non-exact
optimize tree T (optional)
repeat
    pick edge e with lowest cost
    if edge e is exact
        return path of e
    else
        plan path corresponding to e
        update cost of e
        mark e as exact
```

Table 2.4: Procedure of the algorithm for view selection.

```
k ← 0
repeat
    for every vertex q ∈ {q_1, ..., q_m}
        T_new ← T
        replace vertex q in T_new by new IK solution
        update lower bound of corresponding edge in T_new
        if c(T_new) < c(T)
            T ← T_new
            k ← 0
        else
            k ← k + 1
            if k > k_max
                return T
```

Table 2.5: Procedure used to optimize $T$ for the algorithm in Table 2.4.

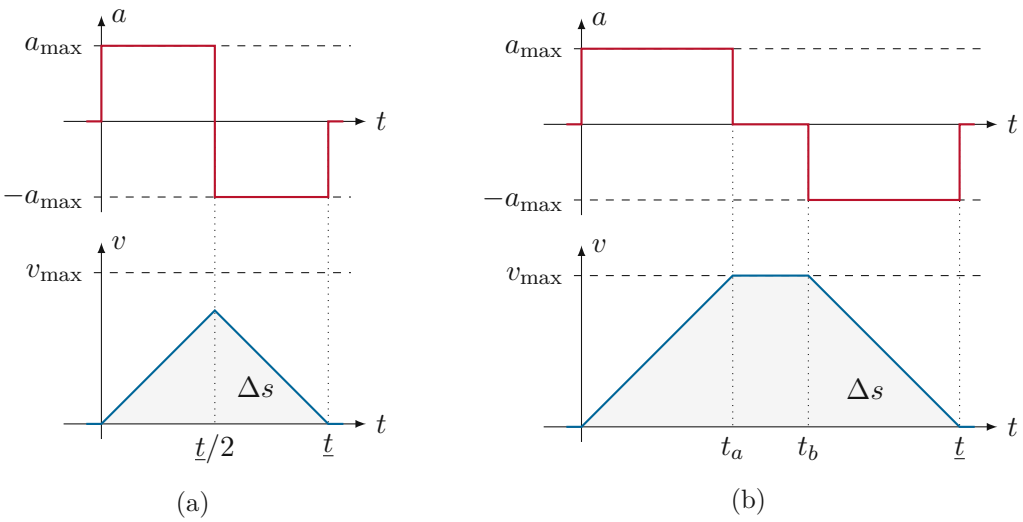



Figure 2.10: Acceleration $a(t)$ and velocity $v(t)$ to travel the distance $\Delta s$ in minimum time $\underline{t}$. (a) Velocity $v$ stays below the limit $v_{\max}$, (b) Velocity limit $v_{\max}$ is attained.

**Single joint.** First, a single joint of the robotic system is considered. All other joints are disregarded for the moment. Let $\Delta s$ denote the absolute difference between the start and goal position of this joint. Figure 2.10 shows the acceleration and velocity profiles to travel this distance in minimal time $\underline{t}$. Depending on the length of $\Delta s$, the velocity limit $v_{\max}$ may be attained or not, see Figure 2.10b and Figure 2.10a, respectively. The traveled distance corresponds to the area under the velocity profile.

In the case of Figure 2.10a, where the velocity limit $v_{\max}$ is not attained, the lower bound $\underline{t}$ is found by calculating

$$\frac{\Delta s}{2} = \int_0^{\underline{t}/2} a_{\max} t \, \mathrm{d}t = \frac{a_{\max} \underline{t}^2}{8}, \tag{2.16}$$

which is equivalent to

$$\underline{t} = 2\sqrt{\frac{\Delta s}{a_{\max}}}. \tag{2.17}$$

Geometric considerations for the case where the velocity limit $v_{\max}$ is attained, illustrated in Figure 2.10b, lead to the expression

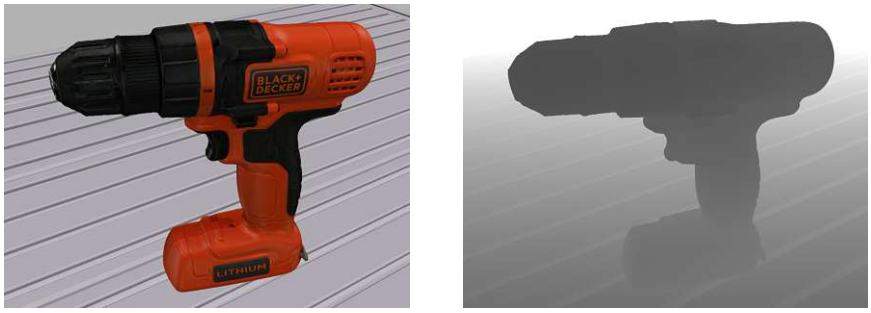
$$\Delta s = (\underline{t} - t_a) v_{\max} = \left(\underline{t} - \frac{v_{\max}}{a_{\max}}\right) v_{\max}, \tag{2.18}$$

which can be reformulated as

$$\underline{t} = \frac{\Delta s}{v_{\max}} + \frac{v_{\max}}{a_{\max}}. \tag{2.19}$$

This yields the lower bound $\underline{t}$ for a single joint

$$\underline{t} = \begin{cases} 2\sqrt{\frac{\Delta s}{a_{\max}}} & \text{if } \Delta s < \frac{v_{\max}^2}{a_{\max}} \\ \frac{\Delta s}{v_{\max}} + \frac{v_{\max}}{a_{\max}} & \text{otherwise.} \end{cases} \tag{2.20}$$



(a) (b)

Figure 2.11: An RGBD image consists of (a) a color image and (b) depth information.

**Multiple joints.** For $n$ joints, the individual lower bounds are denoted by $\underline{t}_i$, $i = 1, \ldots, n$. The joint with the greatest lower bound is the bottleneck and thus determines the total lower bound $\underline{t}_p$ for the path. This is expressed by

$$\underline{t}_p = \max\{\underline{t}_1; \underline{t}_2; \ldots; \underline{t}_n\}. \tag{2.21}$$

## 2.4 Object Reconstruction

The previous sections were concerned with the task of moving the camera to desired views. Now, the transition to the actual object reconstruction is made. To start with, the following text describes how data on the geometric properties of a physical object is obtained. Thereafter, further necessary processing steps are explained. Many illustrations in this and the following sections make use of the YCB object and model dataset [37].

### 2.4.1 Data Acquisition

The data for building a 3D model is acquired using an RGBD camera. Such devices create two pieces of data, a color image (RGB) of the object's texture and a depth image (D), which encodes the distance of the object to the image plane. This is illustrated in Figure 2.11, where darker parts of the depth image correspond to regions that are closer to the image plane.

**Color & Depth.** Although the color image is not a precondition for surface reconstruction, it is nevertheless useful. For example, the algorithm in [38] utilizes the color information for local registration, to be discussed in Section 2.4.2. Each pixel in the depth image encodes the distance of the corresponding 3D point to the image plane. Two important technologies for measuring depth are *time-of-flight* and *structured light*. With the principle of time-of-flight, the depth is calculated based on the time difference between emitting and receiving radiation. In contrast, the method of structured light uses triangulation to infer the depth. A projector illuminates the object with a special pattern whose reflected version is captured by a camera. This approach is pursued by the PHOTONEO MotionCam-3D M, which is used in this work.

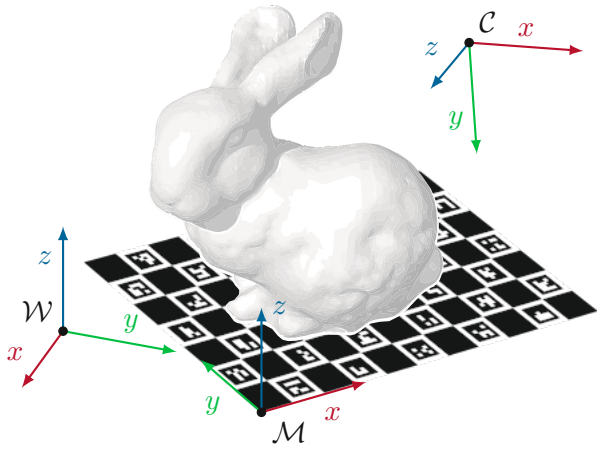



Figure 2.12: Application of fiducial markers for global registration.

**Point Clouds.** A *point cloud* $\mathcal{P}$ is a set of three-dimensional points $\mathbf{p} \in \mathbb{R}^3$. The algorithms applied in this work rely on data in form of point clouds, which is why the RGBD data must first be converted.

### 2.4.2 Scan Registration

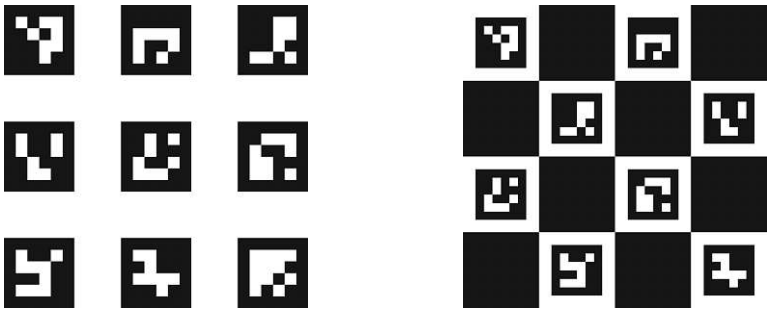
The point clouds acquired from different views are given in their respective camera frame $\mathcal{C}$, corresponding to the pose where the scan was made. To create a full model of a physical object, the scans need to be aligned in a common coordinate frame $\mathcal{M}$. Finding the appropriate transformations $\mathbf{H}^{\mathcal{C}}_{\mathcal{M}}$ to achieve this is a fundamental task in the process of reconstruction, known as *registration*. Ordinarily, the procedure of registration is divided into two steps. The first step is *global registration*, which results in a rough initial alignment of the scans. Afterward, *local registration* tries to refine the initial alignment to increase the quality of the reconstruction.

#### Global Registration

Each joint of the considered robotic system is equipped with a sensor to measure its current position. Due to noise and other inaccuracies, only distorted versions $\tilde{\mathbf{q}}$ of the real joint positions $\mathbf{q}$ are available. Based on these measurements $\tilde{\mathbf{q}}$ and the transformation $\mathbf{H}^{\mathcal{C}}_{\mathcal{W}}(\mathbf{q})$, known from the kinematic model of the robot, an estimate of the camera's pose in the world frame $\mathcal{W}$ can be easily determined by evaluating $\mathbf{H}^{\mathcal{C}}_{\mathcal{W}}(\tilde{\mathbf{q}})$. Applying the corresponding transformation to each point cloud leads to an initial alignment of the scans. However, due to the serial chain of sensors, measurement errors may sum up. Additionally, deviations between the kinematic model of the robot and the real system affect the quality of pose estimation. Such deviations can be mitigated by calibrating the robot. However, the robot used in this thesis is not calibrated, which possibly leads to unsatisfactory results. This is why an alternative approach for global registration is considered next.

(a) (b) (c)

Figure 2.13: Examples of ArUco markers: (a) $4 \times 4$ bits, (b) $5 \times 5$ bits, (c) $6 \times 6$ bits.



(a) (b)

Figure 2.14: Two examples of marker boards: (a) ArUco board, (b) ChArUco board.

By use of *fiducial markers*, the pose of the camera relative to a new fixed coordinate frame, called the marker frame $\mathcal{M}$, can be estimated directly without measuring any joint position. This concept is illustrated in Figure 2.12. The markers are placed near the object in such a way that at least some of them are visible from the camera's view. A special pattern of the markers enables their detection in the RGBD image and subsequently the estimation of $\mathbf{H}^{\mathcal{C}}_{\mathcal{M}}$. Thus, the scans can be aligned in the common marker frame $\mathcal{M}$. An overview of many state-of-the-art marker patterns and their performance is given in [39]. ArUco markers [40] provide great flexibility in generating patterns and are also implemented in OpenCV [41], which is why they are applied in this thesis.

Figure 2.13 depicts some examples of ArUco markers. They are chosen from predefined *dictionaries*, i. e. sets of markers with a common number of bits. In addition to the choice of the dictionary, the side length $l_m$ of the marker can be chosen. It should be large enough such that the marker is clearly visible in the image.

Because of possible occlusions and partial views, it is necessary to use multiple markers. Instead of estimating the pose of each marker individually, the markers can be combined to a whole *board*, also called a *bundle*, which then acts as a single marker for pose estimation. An example of an ArUco board is shown in Figure 2.14a, where the individual markers are arranged in a $3 \times 3$ grid. A variant of an ArUco board, a so-called ChArUco board, is depicted in Figure 2.14b. It is a mixture of a chessboard and ArUco board, hence the name. The corners in the chessboard pattern enable a very accurate pose estimation. Therefore, a ChArUco board will be used in this work.

The marker frame $\mathcal{M}$ associated with a ChArUco board is defined as illustrated in Figure 2.12. Its origin is located at the lower-left corner of the board. The $z$-axis of $\mathcal{M}$ is

perpendicular to the marker board, whereas the $x$- and $y$-axis are aligned with the bottom and left edge, respectively.

The estimation of the transformation $\mathbf{H}_{\mathcal{M}}^{\mathcal{C}}$ by means of a ChArUco board consists of the following steps, which correspond to the procedure shown in Figure 2.15.

1. The starting point is an image where the marker board is visible, depicted in Figure 2.15a. For the purpose of illustration, only the ChArUco board without any object to be scanned is shown. Additionally, the camera matrix and the distortion coefficients of the camera need to be known.

2. Then, the ArUco markers on the board are detected in the image. Figure 2.15b visualizes detected markers with a green border. Due to the partial view, not every marker is detected successfully. For example, the marker on the bottom is not fully visible.

3. Next, corners of the chessboard pattern between two successfully detected markers are interpolated. This is illustrated in Figure 2.15c with green indicators.

4. The last step is to actually estimate the pose of the camera relative to the marker, i.e. the transformation $\mathbf{H}_{\mathcal{M}}^{\mathcal{C}}$. This estimation is based on the interpolated corners and the knowledge that they are arranged in a plane. Figure 2.15d depicts an axis cross corresponding to the estimated pose.

### Local Registration

The aim of local registration is to refine the initial alignment obtained by global registration. Often, a variant of the Iterative Closest Point (ICP) algorithm is used to align a pair of scans, which is discussed first. The extension to locally register the whole set of scans is presented afterward.

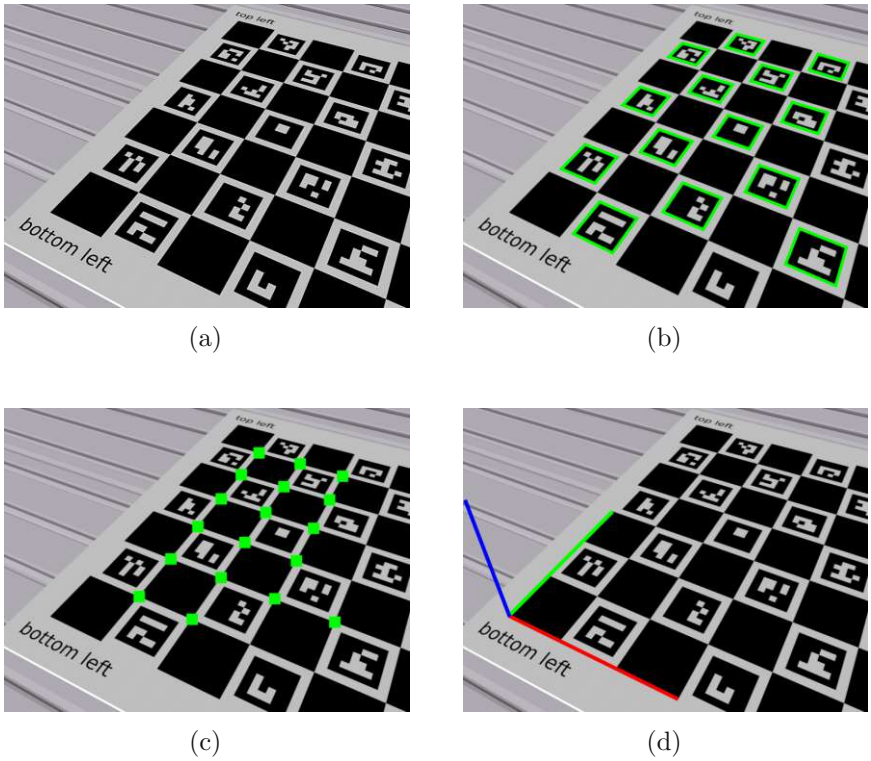
**Pairwise Registration.** For pairwise local registration, two point clouds, the source point cloud $\mathcal{P}_s$ in coordinate frame $\mathcal{S}$ and the target point cloud $\mathcal{P}_t$ in coordinate frame $\mathcal{T}$, are given. The goal is to find a transformation $\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}$ to align $\mathcal{P}_s$ to $\mathcal{P}_t$, starting from an initial guess for $\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}$. An ICP algorithm tackles this problem by iteratively performing two steps:

1. A correspondence set $\mathcal{K} = \{(\mathbf{p}_s, \mathbf{p}_t)\}$ with $\mathbf{p}_s \in \mathcal{P}_s$ and $\mathbf{p}_t \in \mathcal{P}_t$ based on the transformed source point cloud $\mathcal{P}_s(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}})$ and target point cloud $\mathcal{P}_t$ is calculated.

2. Then, an error function $\varepsilon(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}, \mathcal{K})$ is minimized with respect to $\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}$. The found transformation $\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}$ is the starting point for the next iteration.

The error function used by the point-to-point ICP algorithm [42] is

$$\varepsilon(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}, \mathcal{K}) = \sum_{(\mathbf{p}_s, \mathbf{p}_t) \in \mathcal{K}} \left\| \mathbf{p}_t - \mathbf{p}_s(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}) \right\|^2, \tag{2.22}$$




(a) (b)

(c) (d)

Figure 2.15: Steps to estimate the transformation $\mathbf{H}_{\mathcal{M}}^{\mathcal{C}}$: (a) Original image, (b) Detected ArUco markers, (c) Interpolated ChArUco corners, (d) Estimated pose.

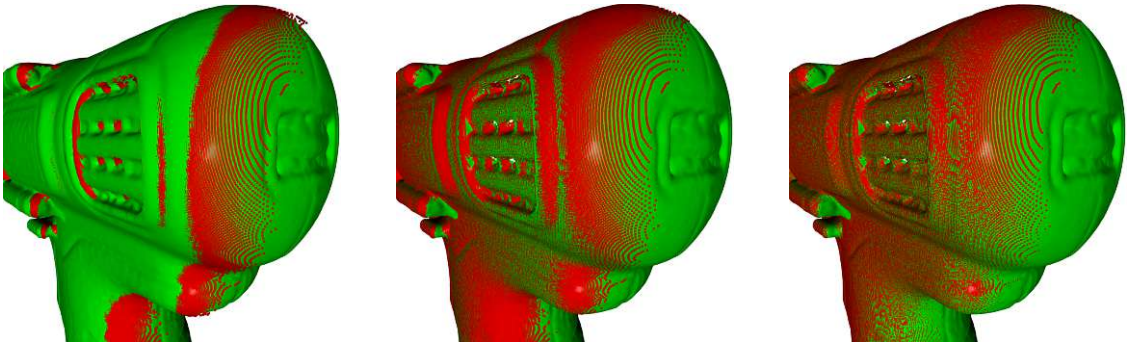
where $\mathbf{p}_s(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}})$ stands for the point $\mathbf{p}_s \in \mathcal{P}_s$, transformed with $\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}$. In contrast, the point-to-plane ICP algorithm [43] requires the normal vector $\mathbf{n}_t$ of each point $\mathbf{p}_t \in \mathcal{P}_t$ to minimize the error function

$$\varepsilon(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}}, \mathcal{K}) = \sum_{(\mathbf{p}_s, \mathbf{p}_t) \in \mathcal{K}} \Big( (\mathbf{p}_t - \mathbf{p}_s(\mathbf{H}_{\mathcal{T}}^{\mathcal{S}})) \cdot \mathbf{n}_t \Big)^2. \tag{2.23}$$

In [44], it is shown that the point-to-plane ICP algorithm converges faster than the point-to-point ICP algorithm. Furthermore, the registration result can be more accurate, as is demonstrated in the following example.

Figure 2.16 shows the alignment of two point clouds, colored in red and green. The initial alignment produced by the pose estimation with a ChArUco board, as discussed previously in Section 2.4.2, is illustrated in Figure 2.16a. It is clearly visible that the two point clouds are not perfectly aligned. The application of the point-to-point ICP algorithm leads to the finer alignment depicted in Figure 2.16b. Compared to that, the point-to-plane ICP algorithm performs noticeably better, which can be seen in Figure 2.16c. Therefore, the point-to-plane ICP algorithm will be used in this work.

**Full Registration.** With the ICP algorithms discussed above, a pair of point clouds is registered. However, in general, a set of more than two scans must be aligned to each



(a) (b) (c)

Figure 2.16: Illustration of local registration: (a) Initial alignment after global registration and results of (b) point-to-point and (c) point-to-plane ICP registration.

other. The applied method for registering the whole set of scans is based on the approach of [45], which uses ICP registration in combination with pose graph optimization. In this context, the vertices of the pose graph correspond to point clouds and the edges represent transformations between them. The applied algorithm assumes a classification of edges into *odometry edges* and *loop closure edges*. In this work, the classification is done as follows.

The size of the correspondence set $\mathcal{K}$ is calculated for each pair of globally registered point clouds. Figure 2.17 illustrates an example with five point clouds $\mathcal{P}_0, \ldots, \mathcal{P}_4$ where the weights of the edges are set to the respective size $|\mathcal{K}|$. Because of sequence optimization and view planning, it is not guaranteed that consecutive point clouds have a large overlap. Hence, a maximum spanning tree is calculated to determine point clouds with maximum possible overlap. The edges contained in the maximum spanning tree are considered odometry edges, whereas all other edges are classified as loop closure edges.

Then, each edge of the pose graph is initialized with a transformation. For odometry edges, the initial transformation is obtained by pairwise registration of the corresponding point clouds with the point-to-plane ICP algorithm. In contrast, loop closure edges are initialized with the identity transformation, i. e. the relative alignment of the corresponding point clouds remains as computed in the global registration step. Finally, pose graph optimization is performed with the Levenberg–Marquardt algorithm to determine the ultimate transformations for local registration.

### 2.4.3 Surface Reconstruction

The result of the registration step discussed previously is a set of aligned point clouds, which needs to be integrated for further processing. To this end, the point clouds are first unified and then down-sampled. The down-sampling is done using a regular voxel grid to reduce the point density in overlapping regions.

After the integration step, the obtained point cloud is prepared for surface reconstruction. Algorithms for this purpose have to deal with imperfections, for instance noise and missing data [46]. The Poisson surface reconstruction method [47] is capable of

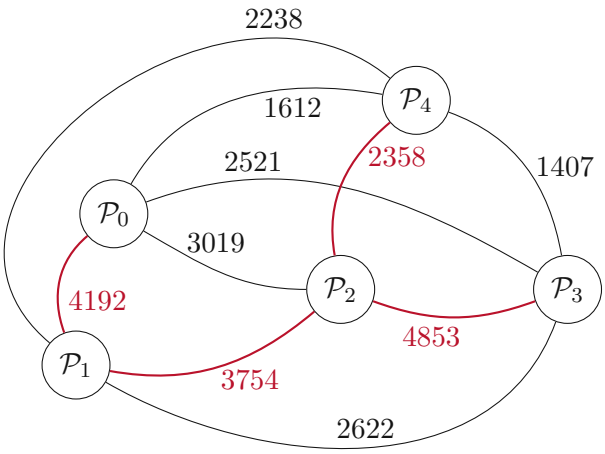



Figure 2.17: Example of a pose graph with five point clouds. A maximum spanning tree constituting the odometry edges is highlighted. All other edges are considered loop closure edges.

interpolating data in regions where no points are available. In contrast to other algorithms, for example the ball-pivoting algorithm [48], it produces smooth surfaces by solving a regularized optimization problem. This motivates the application of the Poisson surface reconstruction method in this thesis.

# 3 Implementation

Based on the discussion of the underlying concepts in Chapter 2, this chapter gives an overview of the reconstruction system's implementation. The existing infrastructure to execute trajectories on the robotic system at TU Wien uses the Robot Operating System (ROS) [49], more specifically the distribution *Melodic Morenia*. ROS is an open-source project and offers software libraries to build robot applications. Thus, this platform is also utilized for this thesis. It facilitates the communication between programs by means of *messages* and *services*. Messages are a form of asynchronous communication, whereas services are based on a request-response scheme. Furthermore, a *parameter server* is available to share parameters among programs.

The organization of the reconstruction system implemented in ROS is shown in Figure 3.1. Arrows represent communication channels based on the three schemes mentioned before. Building blocks belonging to each other are consistently colored. This leads to four groups, namely *motion planning*, *object reconstruction*, *user interface*, and *automatic mode*. In this order, the following sections describe the functionality of the components within these groups in more detail.

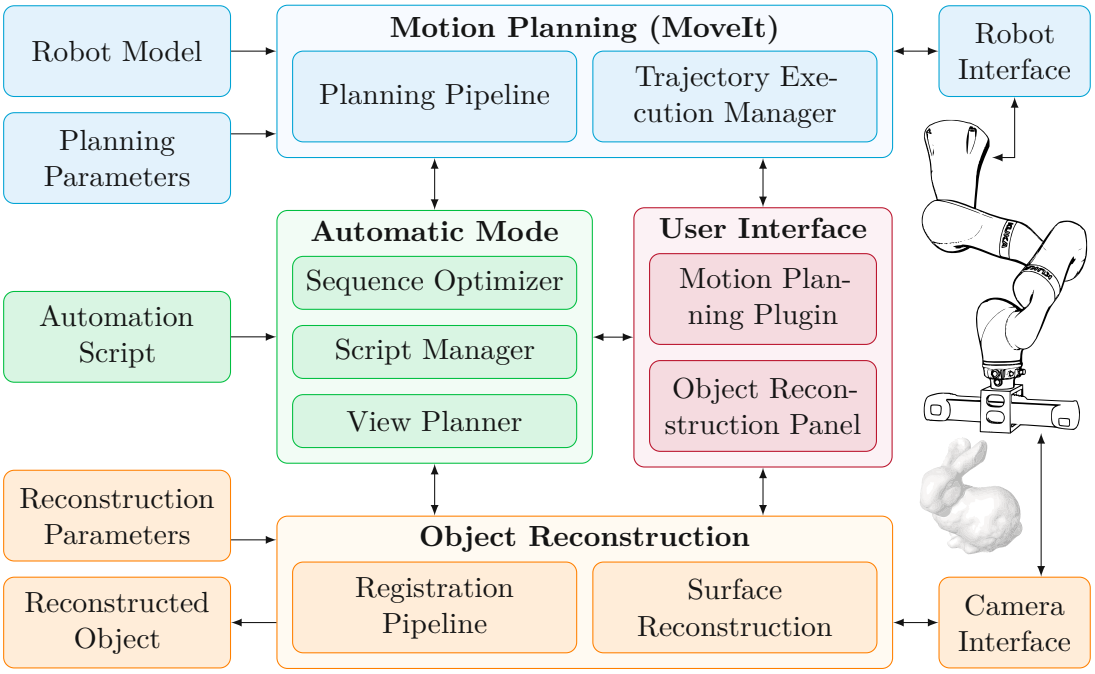



Figure 3.1: Overview of the reconstruction system's implementation.

## 3.1 Motion Planning

To facilitate the application of existing motion planning algorithms and to aid research in this field, software packages for motion planning have been developed. One example is the OpenRAVE project [50], which offers all required components to implement and solve specific motion planning problems. In contrast, another approach is taken by the Open Motion Planning Library (OMPL) [51]. It focuses on sampling-based algorithms and makes a clear separation to other necessary functionality like collision checking. This design choice makes it easy to integrate OMPL with other front-ends and libraries and led to wide usage.

MoveIt [52], an open-source project initially developed at Willow Garage, is a motion planning framework that integrates OMPL and uses it as its standard motion planning library. A large community actively contributes to MoveIt and it has been used on over 126 robots. It provides motion planning capabilities to ROS and is therefore ideally suited for this thesis. The following paragraphs refer to the upper part of Figure 3.1 and describe the main components of MoveIt along with the steps needed to implement the considered robotic system. Further information about MoveIt can be found, for example, in [53].

**Robot Model.** The model of the robotic system for motion planning involves the kinematic and geometric properties of the links and joints. On startup, MoveIt loads this information in form of a Unified Robot Description Format (URDF) file [54], which uses XML syntax to describe the robot. In this work, a URDF file incorporating frame, linear axes, collaborative robot, depth camera, and table was created, see Figure 3.2. To each visual object in this figure, there is an associated collision object of simplified shape as explained in Section 2.2.4. Additionally, MoveIt uses a second file, called the Semantic Robot Description Format (SRDF) file [55], to complement the robot model with planning information. For example, the joints for which a motion plan is computed are listed in it. This file is automatically generated at the end of the configuration in MoveIt's setup assistant.

**Planning Parameters.** In several configuration files, planning parameters for MoveIt can be adjusted. Examples are the default motion planner or soft limits for the joints. As will be motivated in Section 4.1, the motion planner deployed in this thesis is the RRT-connect algorithm [56]. Soft limits for the revolute joints are set to 10°.

**Planning Pipeline and Trajectory Execution Manager.** A core component of MoveIt is the planning pipeline. It performs collision checking and accesses the planning library, which is OMPL by default. Other libraries can be used by means of plugins. Planned paths are time-parameterized in a postprocessing step and handed over to the trajectory execution manager, also belonging to MoveIt. For this thesis, the time-parameterization method was changed to the *time-optimal trajectory generation algorithm* [57]. This algorithm outputs differentiable trajectories, which are needed for execution on the real system.

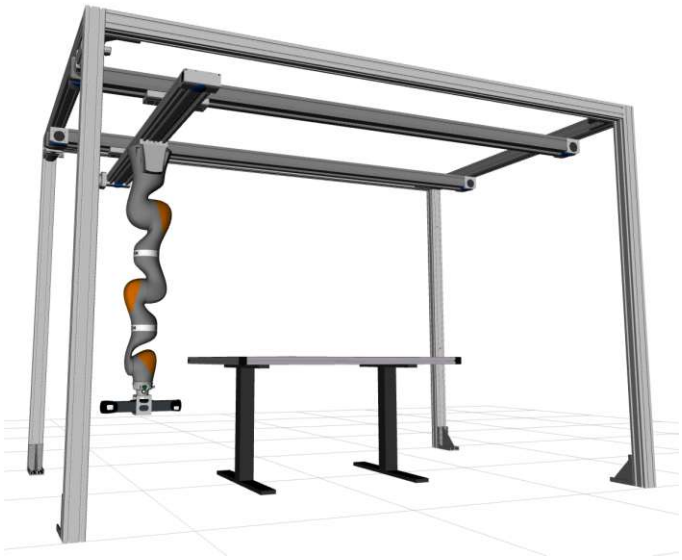


Figure 3.2: Visualization of the robotic system including the linear axes. It is defined by means of a URDF file to be used in MoveIt.

**Robot Interface.** The trajectory planned in MoveIt is sent to the robot interface in form of a ROS message. Instead of using the real robotic system, the Gazebo simulation environment [58] may be deployed. It is able to interface with ROS and thus allows an almost seamless transition between real hardware and simulation. To use it in this context, the URDF file was extended with properties regarding the dynamics of the system, e. g. the mass and moments of inertia of each link.

## 3.2 Object Reconstruction

In this section, the lower part of Figure 3.1, responsible for object reconstruction, is explained. In contrast to the motion planning part, which is primarily based on MoveIt, the object reconstruction is self-developed and deploys the software libraries OpenCV [41] and Open3D [59]. These are powerful open-source projects that implement numerous algorithms related to computer vision and object reconstruction.

**Camera Interface.** When a scan is triggered, the reconstruction system obtains the RGBD image data from the camera interface. This component depends on whether real hardware or Gazebo is used. In the first case, the API of Photoneo is deployed to directly fetch data from the Photoneo MotionCam-3D M. Otherwise, i. e. when a simulation is performed in Gazebo, the data is generated by a depth camera plugin. This enables the possibility to scan objects virtually. Hence, the whole developed reconstruction system can be operated in simulation, which is a great advantage when testing possible future extensions.

**Registration Pipeline.** The input to the registration pipeline is an RGBD image obtained from the camera interface. First, the camera pose is estimated with OpenCV based on a ChArUco board, according to the description in Section 2.4.2. If this way of pose estimation fails, e. g. when no markers are detected, an estimate based on the measured joint positions is computed. The following processing steps make use of Open3D, starting with the conversion of the RGBD image to a point cloud. The point cloud is then transformed with the estimated pose for global registration. To remove undesired components contained in the point cloud, e. g. the surface of the table, the point cloud is automatically cropped with a bounding box. This bounding box is defined before the reconstruction process starts (see also Section 3.3). Multiple scans are locally registered based on pose graph optimization as discussed in Section 2.4.2.

**Surface Reconstruction and Reconstructed Object.** The last step is to fuse the scans and reconstruct the object's surface, also utilizing Open3D's functionality. To this end, the point clouds are combined and down-sampled. Consistently oriented normal vectors are computed and the Poisson surface reconstruction algorithm is applied as explained in Section 2.4.3. This way, a triangle mesh of the scanned object is obtained, which can be saved on the computer in PLY file format.

**Reconstruction Parameters.** It is possible to specify the used marker board and other processing parameters in configuration files. These files are loaded by the reconstruction system on startup.

## 3.3 User Interface

The user interface of the reconstruction system is based on Rviz [60], a visualization application for ROS. As illustrated in the middle of Figure 3.1, the user interface is composed of MoveIt's motion planning plugin and a newly developed Rviz panel for object reconstruction, called the *object reconstruction panel*. Therein, it is possible to access the functionality of the reconstruction system. The object reconstruction panel consists of several tabs that are explained in the following.

**Plan.** By means of the plan tab, depicted in Figure 3.3, the camera can be positioned. To this end, the representation of poses as introduced in Section 2.1.2 is applied. Hence, the desired camera pose is determined by adjusting the focus point $\mathbf{f}$, radius $r$, azimuth angle $\varphi$, polar angle $\vartheta$, and angle $\alpha$. It is possible to specify the camera pose relative to the world frame $\mathcal{W}$ or relative to the marker frame $\mathcal{M}$. Once a pose is defined, a path can be planned and executed by switching to the motion planning plugin of MoveIt. Moreover, the plan tab allows adding a bounding box. This bounding box is set to avoid collisions with the object to be scanned. Additionally, it is used to crop the obtained point cloud, i. e. to separate the object from the environment.

**Scan and Register.** Figure 3.4a shows the scan tab of the object reconstruction panel. It allows to connect to the depth camera and to trigger scans manually. Furthermore,

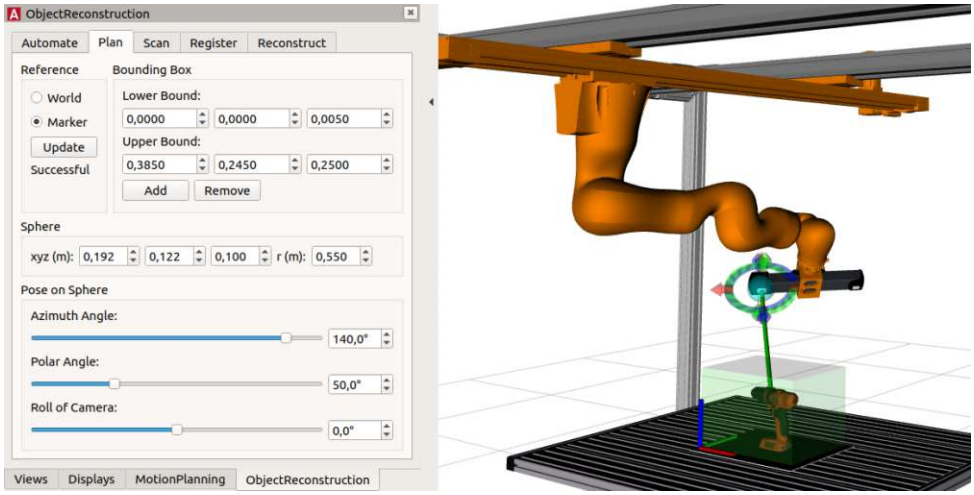
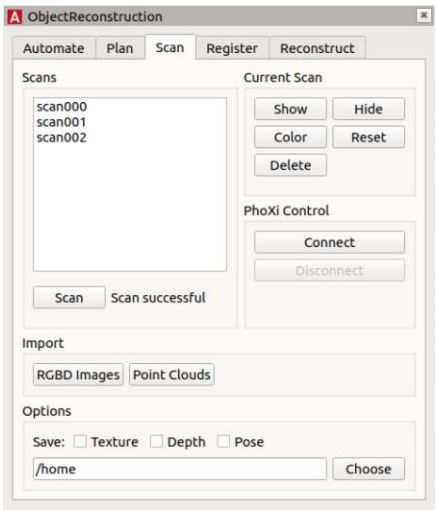
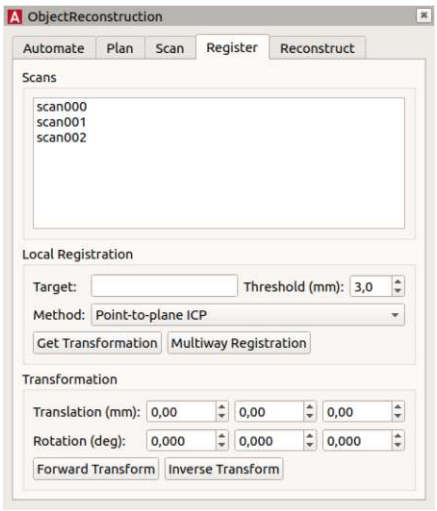



Figure 3.3: Plan tab of the object reconstruction panel in Rviz.

scans can be displayed and colored for visualization. Within the register tab, illustrated in Figure 3.4b, scans can be registered with the local registration methods discussed in Section 2.4.2. If pairwise registration is desired, the source point cloud has to be selected first. Then, the transformation is obtained by defining a target point cloud and clicking on *Get Transformation*. Moreover, it is possible to perform a full registration by clicking on *Multiway Registration*.

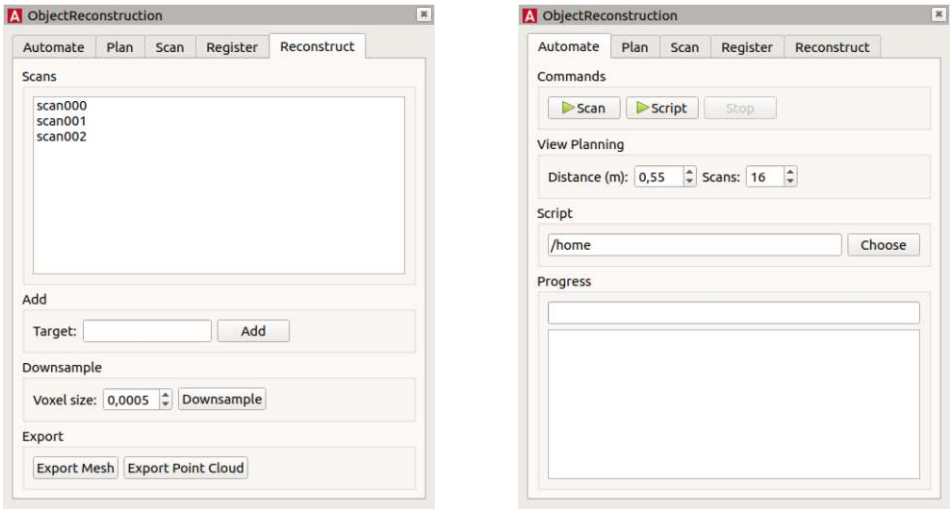



(a) Scan tab.




(b) Register tab.

Figure 3.4: Tabs of the object reconstruction panel for (a) connecting to the depth camera and triggering scans and for (b) registering the fragments.




(a) Reconstruct tab. (b) Automate tab.

Figure 3.5: Tabs of the object reconstruction panel for (a) exporting the reconstruction and for (b) starting an autonomous scan or a script.

**Reconstruct and Automate.** Registered scans can be combined and down-sampled in the tab for reconstruction, see Figure 3.5a. This tab is also used to export point clouds and to export a mesh, i. e. to start the Poisson surface reconstruction algorithm.

Furthermore, the user interface allows starting a whole reconstruction process instead of triggering commands manually (see also Section 3.4). For this purpose, the automate tab shown in Figure 3.5b is accessed. There are two possibilities to start a reconstruction process. On the one hand, it is possible to choose a script, which contains predefined views. This corresponds to the model-based mode introduced at the beginning of Chapter 2. Alternatively, an autonomous reconstruction is performed with the non-model-based mode. This mode deploys the view planning algorithm explained in Section 2.1.3. To this end, the robot has to be moved to a position for the initial scan. Then, the autonomous reconstruction is started by specifying the distance $d_c$ of the camera to the estimated ellipsoid and the desired number of scans.

## 3.4 Automatic Mode

The purpose of the self-developed automatic mode is to be able to perform an object reconstruction with minimal user interaction. This part of the system interfaces with the motion planning and object reconstruction functionalities in the same way as the user interface by utilizing ROS messages and services. The components belonging to the automatic mode are highlighted in green, see Figure 3.1.

**Automation Script and Script Manager.** For the model-based mode, desired views are specified in advance. To this end, *automation scripts* are used, which are files written in XML syntax. Elements for moving the robot, triggering scans, registering scans,

etc. were defined in an XML Schema file. Automation scripts allow great flexibility because commands can be executed in arbitrary order. Furthermore, it is possible to use Xacro [61] to define custom macros. The *script manager* parses an automation script, validates it, and delegates the included commands to the motion planning and object reconstruction components.

**Sequence Optimizer.** Optionally, the loaded automation script may be preprocessed by the *sequence optimizer* before the reconstruction process starts. The sequence optimizer is implemented based on the discussion in Section 2.3 and hands over the optimized sequence as well as the planned paths to the script manager.

**View Planner.** The *view planner* is relevant for the non-model-based mode, i. e. to perform an autonomous reconstruction. This ROS node implements the algorithm proposed in Section 2.1.3. Given the desired number of scans and the distance $d_c$ of the camera to the ellipsoid's surface, it starts with an initial scan. To estimate the ellipsoid in each iteration, a bounding box of the partially observed object is calculated in Open3D. The dimensions of this bounding box are then scaled by a factor of $\sqrt{3}$ to obtain the extent of the desired ellipsoid. Due to this scaling, the ellipsoid encompasses the bounding box and thus the partially observed object. To select the next best view, the view planner accesses the services offered by the sequence optimizer, see also Section 2.3.3.

# 4 Simulations

This chapter presents simulation studies of the developed reconstruction system. In Section 4.1, various motion planning algorithms are tested for a typical planning request. A suitable algorithm is identified, which is then used to plan motions for all further simulations and experiments. Next, a reconstruction of an object with given views is performed in Section 4.2. This is to analyze aspects of registration and surface reconstruction. Subsequently, Section 4.3 revisits the scenario of Section 4.2 to demonstrate the capabilities of the implemented sequence optimization. Finally, the proposed view planning algorithm is applied in Section 4.4. This serves to illustrate an autonomous object reconstruction.

## 4.1 Motion Planning Algorithms

The goal of this section is to identify a sampling-based motion planning algorithm that is well suited for the considered robotic system. To this end, the benchmarking facilities of MoveIt are used [62], which provide an infrastructure to define planning problems, run algorithms, and compare their performance.

### 4.1.1 Benchmark Problem

Figure 4.1 shows the planning problem to be solved. The joint positions of the robot's start state (colored in green) and its goal state (colored in orange) are given in Table 4.1. A cube of side length 0.3 m is placed on top of the table as a collision object. Such a scenario, where the robot has to find its way to the other side of the table, occurs frequently during the reconstruction of objects.

OMPL is MoveIt's standard planning library and comprises the most common sampling-based algorithms. At the time of writing, 23 algorithms of OMPL are available in MoveIt, which are all applied to the same planning problem mentioned before. The allowed planning time is set to 5 s. If this timeout is exceeded, the planning problem is considered not solved.

| Joint | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $q_8$ | $q_9$ |
|---|---|---|---|---|---|---|---|---|---|
| Start state | −1 m | 0 m | 0° | 0° | 0° | 0° | 0° | 0° | 0° |
| Goal state | 0.01 m | −0.33 m | −86° | −93° | −37° | −80° | −82° | −49° | −116° |

Table 4.1: Joint positions of the robot's start and goal state for benchmarking motion planning algorithms.

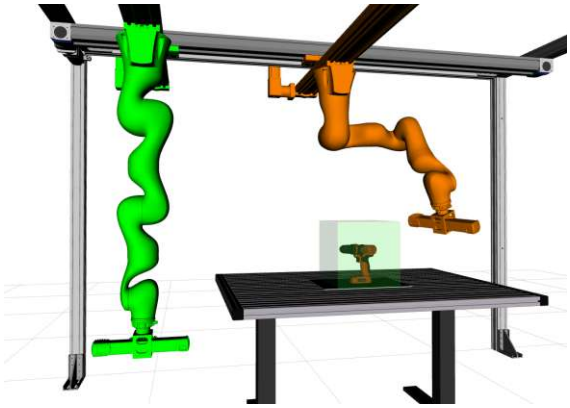


Figure 4.1: Scenario for benchmarking various motion planning algorithms. Start and goal state are colored in green and orange, respectively. A collision object is placed on the table.

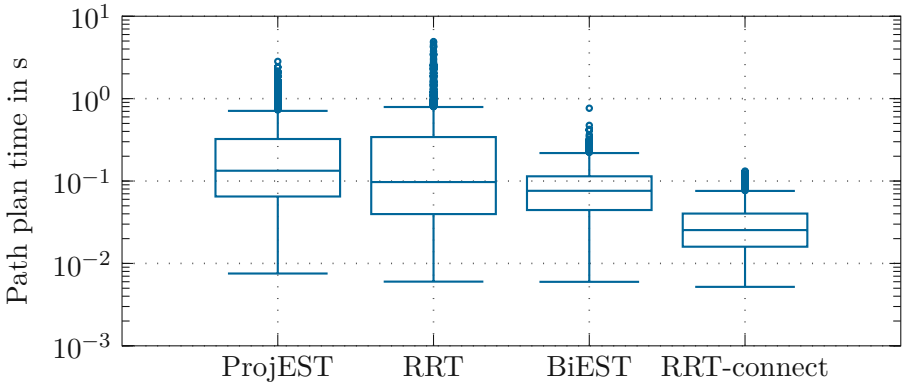



Figure 4.2: Statistics on the path plan time of four motion planning algorithms for the benchmark scenario depicted in Figure 4.1.

### 4.1.2 Comparison

An important performance criterion of a planning algorithm is its ability to quickly solve a planning problem, given a solution exists. Based on this criterion, the following evaluation focuses on four algorithms that performed best in this benchmark, namely the ProjEST [63], RRT [64], BiEST [63], and RRT-connect [56] algorithm. Other criteria for comparing planning algorithms are, for example, the length or the smoothness of the found solution.

The computation of 1000 planning requests per algorithm was performed with an 8×1.8 GHz Intel Core i7 and 8 GB of RAM. In Figure 4.2, box plots of the path plan time are shown on a logarithmic scale. The algorithms BiEST and RRT-connect performed noticeably better than ProjEST and RRT. For the RRT algorithm, the longer path plan time also decreases the rate of success to 98.5 %. In contrast, the ProjEST, the BiEST, and the RRT-connect algorithm were able to solve all planning requests successfully. Based on

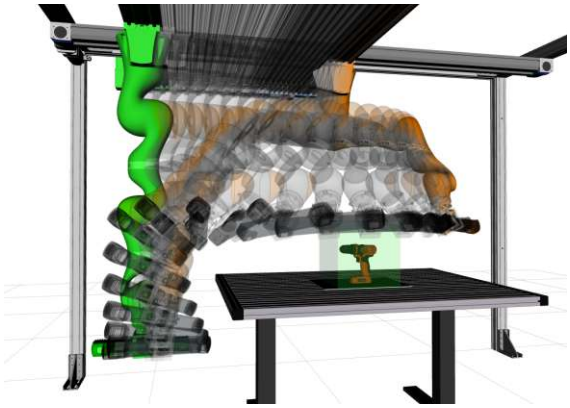


Figure 4.3: A path from start configuration (green) to goal configuration (orange) planned by the RRT-connect algorithm.

these results, the RRT-connect algorithm is selected for the following simulations and all further experiments.

A path planned with the RRT-connect algorithm for this benchmark scenario is illustrated in Figure 4.3. Note that MoveIt also performs some post-processing steps to simplify and smooth the path computed by the RRT-connect algorithm. The time-parameterized version of the processed path, i. e. the obtained trajectory, is depicted in Figure 4.4.

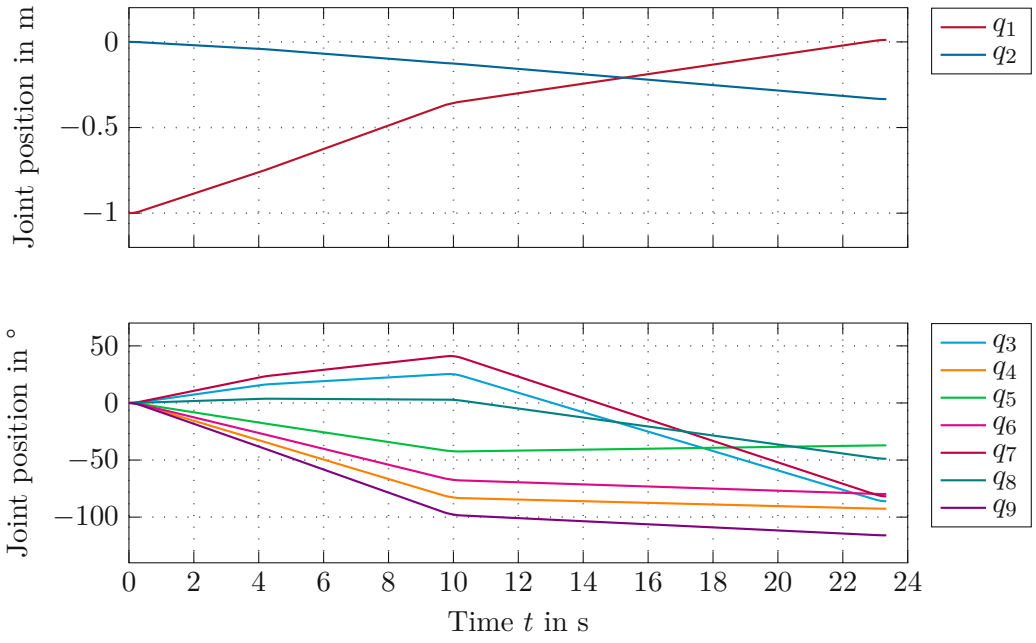



Figure 4.4: Trajectory obtained by time-parameterizing the path illustrated in Figure 4.3.

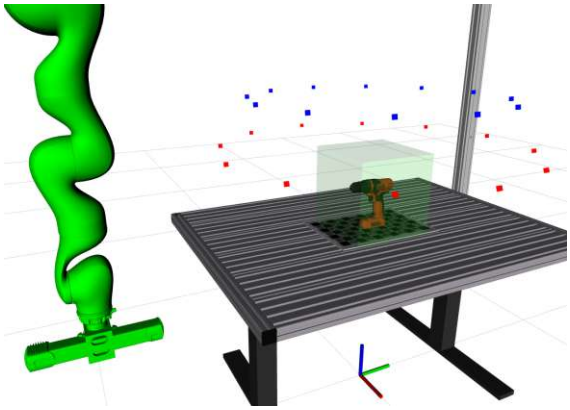


Figure 4.5: Scenario for reconstructing an object with 24 scans. Views $1-12$ and $13-24$ are illustrated as red and blue points, respectively. The start configuration is colored in green. A collision object is placed on the table.

## 4.2 Object Reconstruction

The quality of the obtained 3D model is strongly influenced by the alignment of the scans. This motivates the following simulation, where the methods of Section 2.4.2 and Section 2.4.3 are demonstrated. In particular, the performance of global registration with a ChArUco board is analyzed. Furthermore, some characteristics of the surface reconstruction are illustrated.

### 4.2.1 Scenario

The scenario for analyzing the quality of registration is illustrated in Figure 4.5. The start configuration is again colored in green and given in Table 4.1. Gazebo is deployed to place the object, in this case a power drill of the YCB model set [37], on the table and simulate the depth camera by means of a plugin. The green cube of side length 0.3 m on the table is used as a collision object for the motion planner as well as a bounding box to crop the obtained point cloud. This way, the power drill can be separated from the rest of the environment.

The task is to move the camera to 24 pose goals $i = 1, \dots, 24$ arranged around the object. They are given in the world frame $\mathcal{W}$ by $\mathbf{f}_i = \begin{bmatrix} 0 & 0 & 0.78\,\mathrm{m} \end{bmatrix}^{\mathrm{T}}$, $r_i = 0.6\,\mathrm{m}$, $\alpha_i = 0$, and the angles

$$\varphi_i = 30°(i-1), \quad i = 1, \dots, 24 \tag{4.1a}$$

$$\vartheta_i = \begin{cases} 70° & \text{for } i = 1, \dots, 12 \\ 50° & \text{for } i = 13, \dots, 24. \end{cases} \tag{4.1b}$$

This representation of pose goals was explained in Section 2.1.2. The expressions in (4.1) constitute two full cycles around the object in steps of 30° with two different polar angles.

| Scans 1 to 12 | | | Scans 13 to 24 | | |
|---|---|---|---|---|---|
| Scan | $d$ / mm | $\theta$ / ° | Scan | $d$ / mm | $\theta$ / ° |
| 1 | 1.638 | 0.087 38 | 13 | 1.595 | 0.074 63 |
| 2 | 1.486 | 0.087 48 | 14 | 1.493 | 0.081 89 |
| 3 | 1.411 | 0.096 66 | 15 | 1.549 | 0.086 82 |
| 4 | 1.445 | 0.086 66 | 16 | 1.592 | 0.090 64 |
| 5 | 1.428 | 0.091 53 | 17 | 1.575 | 0.079 99 |
| 6 | 1.511 | 0.087 65 | 18 | 1.603 | 0.077 24 |
| 7 | 1.667 | 0.088 19 | 19 | 1.594 | 0.078 66 |
| 8 | 1.510 | 0.085 25 | 20 | 1.618 | 0.072 44 |
| 9 | 1.448 | 0.089 96 | 21 | 1.617 | 0.072 27 |
| 10 | 1.395 | 0.096 36 | 22 | 1.594 | 0.091 05 |
| 11 | 1.434 | 0.094 85 | 23 | 1.522 | 0.090 00 |
| 12 | 1.488 | 0.084 69 | 24 | 1.564 | 0.076 07 |

Table 4.2: Translation error $d$ and rotation error $\theta$ between actual camera pose and the estimate obtained with a ChArUco board.

A ChArUco board composed of $11 \times 7$ squares is used to estimate the camera pose. The squares of the ChArUco board are 35 mm long and the ArUco markers within the board have a side length of 25 mm. With this design, the marker board fits on an A3 paper and can be easily printed, which is relevant for the experiments in Chapter 5. For this simulation, the ChArUco board was placed in Gazebo and a calibration of the virtual depth camera was performed prior to the 24 scans.

### 4.2.2 Evaluation

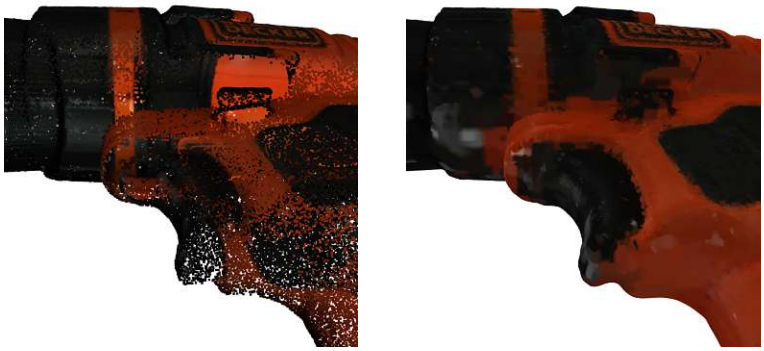
Because the ground truth is available in simulation, the error between the actual camera pose and the one estimated with the ChArUco board can be calculated. In the following, the error is expressed in terms of two numbers. The first one is the Euclidean distance $d$ between the origins of the estimated and actual camera frame, i. e. it represents the translative part. The second quantity is the angle $\theta$ corresponding to an axis-angle representation of the rotation error. Table 4.2 lists these quantities for all 24 scans. On average, the estimate is only shifted by 1.532 mm and rotated by 0.0853°.

The effect of this estimation error on the point cloud is shown in Figure 4.6a. A good initial alignment is achieved, but distortions are visible. After local registration, the point cloud in Figure 4.6b is obtained, where the alignment is noticeably refined. For example, the writings on the power drill are legible, which is not the case in Figure 4.6a. This shows that the applied registration methods are well suited to align the scans.

Some parts of the power drill are not captured, depicted in Figure 4.7a. This figure shows regions of the power drill that are difficult to scan because of geometric limitations. However, if missing areas are small enough, the Poisson surface reconstruction algorithm is able to interpolate incomplete data, see Figure 4.7b.


(a) Globally registered point. (b) Locally registered point.

Figure 4.6: The good initial alignment obtained by pose estimation with the ChArUco board is refined by local registration with the point-to-plane ICP algorithm.



(a) Point cloud. (b) Reconstructed surface.

Figure 4.7: Interpolation effect of Poisson surface reconstruction.

## 4.3 Sequence Optimization

If multiple pose goals are given in advance, sequence optimization can be applied. In this section, the performance of sequence optimization is compared to the case without sequence optimization in terms of the total duration of the trajectory, denoted by $t_p$. This quantity comprises the duration of each individual trajectory between a pair of pose goals, but not the time required to trigger and process scans.

### 4.3.1 Scenario

The sequence optimization is tested for the same scenario as in Section 4.2, i. e. 24 scans are performed at the poses defined in (4.1). Figure 4.5 depicts this scenario.

### 4.3.2 Evaluation

The following discussion considers three cases. In case A, no sequence optimization is performed, i. e. the trajectory is planned for the initial order of the pose goals given in (4.1). Case B is based on sequence optimization with parameter $\beta = 1.5$ (see Table 2.2 in Section 2.3.2), but without the graph optimization according to Table 2.3. This preprocessing step is then added in case C with parameter $k_{\max} = 120$.

Figure 4.8 depicts the empirical cumulative distribution function $F(t_p)$ of the total trajectory duration $t_p$ for each of the three cases mentioned before. This diagram is based on 500 runs per case and shows the performance gain achieved with sequence optimization and the preprocessing step described in Section 2.3.2. The duration $t_p$ roughly ranges from 80 s to 220 s. One reason for this great variation is that the pose goals are arranged densely around the object. Thus, the duration $t_p$ is mainly influenced by situations where the robot has to "unwrap" itself to get out of joint limits, which takes much longer than a direct movement. The goal of sequence optimization is to reorder the poses such that these situations occur less often.

Table 4.3 summarizes the average duration $\bar{t}_p$ and the average number of planning requests $\overline{n}_{\text{req}}$ obtained in this simulation. For case A, $\overline{n}_{\text{req}} = 24$ because the trajectory is planned sequentially for all 24 pose goals. With sequence optimization, this number slightly increases to 27.40 due to $\beta = 1.5$. By decreasing $\beta$, the average duration $\bar{t}_p$ can be further reduced at the cost of more planning requests.

This simulation demonstrates that the average duration $\bar{t}_p$ for this benchmark scenario can be reduced by 10.18 % for case B with little overhead. With the heuristic for graph

| Case | $\bar{t}_p$ | $\overline{n}_{\text{req}}$ |
|---|---|---|
| A | 141.81 s | 24.00 |
| B | 127.37 s | 27.40 |
| C | 120.33 s | 26.99 |

Table 4.3: Summary of the average duration $\bar{t}_p$ and the average number of planning requests $\overline{n}_{\text{req}}$ for all three cases.

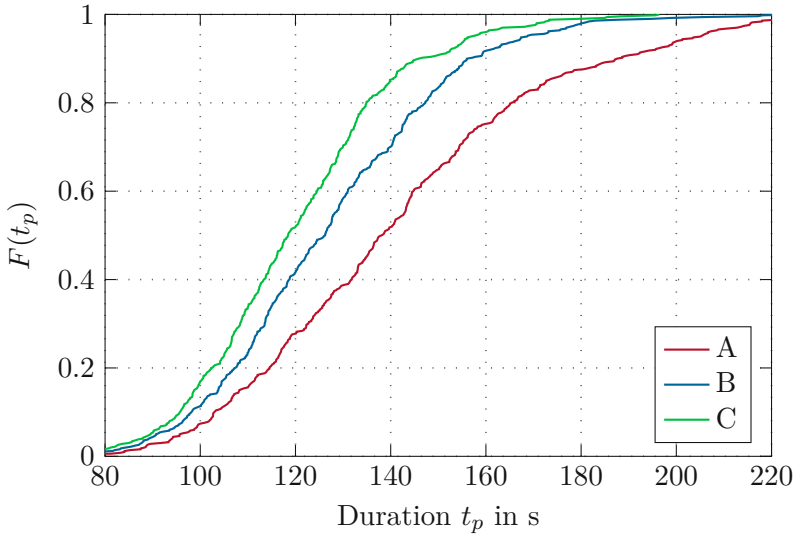



Figure 4.8: Empirical cumulative distribution function of the trajectory's duration $t_p$ for three cases: (A) no sequence optimization, (B) sequence optimization without graph optimization and (C) sequence optimization with graph optimization.

optimization described in Section 2.3.2, an additional performance gain is achieved. Here, the average duration $\bar{t}_p$ is reduced by 15.15 % compared to case A. One has to consider that the initial pose goals were already given in a "natural" order. In other cases, where an unsorted list of pose goals is given, sequence optimization is even more useful.

## 4.4 View Planning

In Section 4.2 and Section 4.3, views were defined in advance to perform an object reconstruction. This is applicable, for instance, to perform an inspection of known objects. Next, the proposed view planning algorithm of Section 2.1.3 is demonstrated in simulation. With this algorithm, it is possible to scan unknown objects autonomously. This eliminates the need to specify views manually. The algorithm only requires an initial scan to start reconstruction.

### 4.4.1 Scenario

The scenario for reconstruction is depicted in Figure 4.9. The configuration of the robot for the initial scan is colored in green. A virtual collision box is placed on the table to avoid collisions between robot and object. The task for the view planning algorithm is to scan the unknown object with 16 scans at a distance of $d_c = 0.55\,\text{m}$ to the estimated ellipsoid.

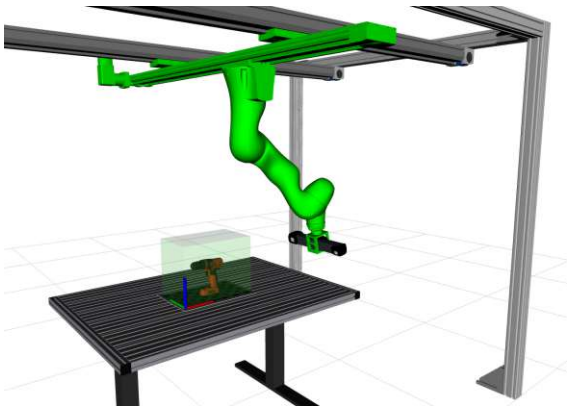
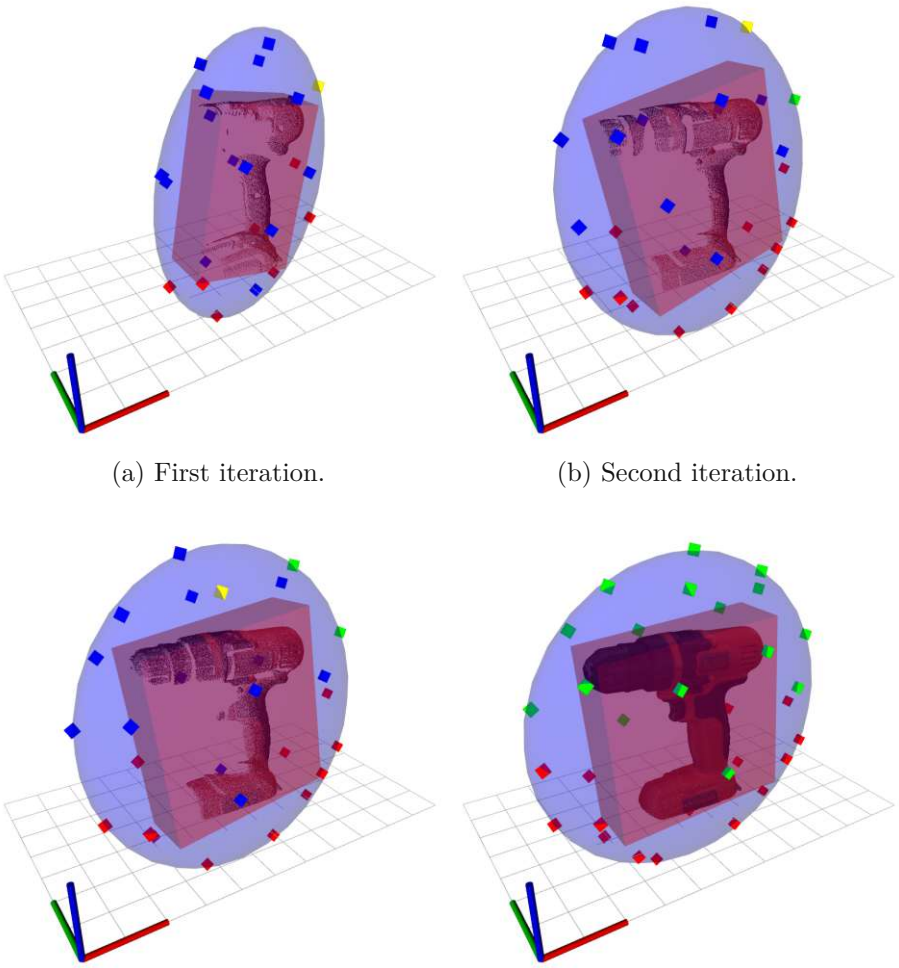


Figure 4.9: Scenario for demonstrating the proposed view planning algorithm. The configuration for the initial scan is colored in green. A collision object is placed on the table to avoid collisions with the object to be scanned.

### 4.4.2 Evaluation

Figure 4.10 shows selected iterations of the reconstruction procedure with the partially scanned object. The estimated bounding box and the overlaid ellipsoid with distributed views on its surface are depicted. For this illustration, the color scheme of Figure 2.5 is used to distinguish infeasible, completed, and planned views as well as the active view, i. e. they are respectively colored in red, green, blue, and yellow. In addition, the grid of the marker board is illustrated for easier size comparison, together with the corresponding marker frame $\mathcal{M}$ (see also Figure 2.12).

At the beginning of the first iteration, shown in Figure 4.10a, only the back part of the power drill is visible. Thus, the initial ellipsoid does not enclose the whole object. In the second iteration, depicted in Figure 4.10b, a very good geometric approximation of the object is already obtained. Hence, the ellipsoid changes only marginally when proceeding in the third iteration, see Figure 4.10c. The final ellipsoid with the completed views is illustrated in Figure 4.10d. A good distribution of views is achieved. Due to the great redundancy of the mechanical setup with linear axes, it is possible to scan the object from a wide variety of views. Infeasible views occur only at the bottom of the object, where the camera would collide with the table.



(a) First iteration. (b) Second iteration.

(c) Third iteration. (d) Last iteration.

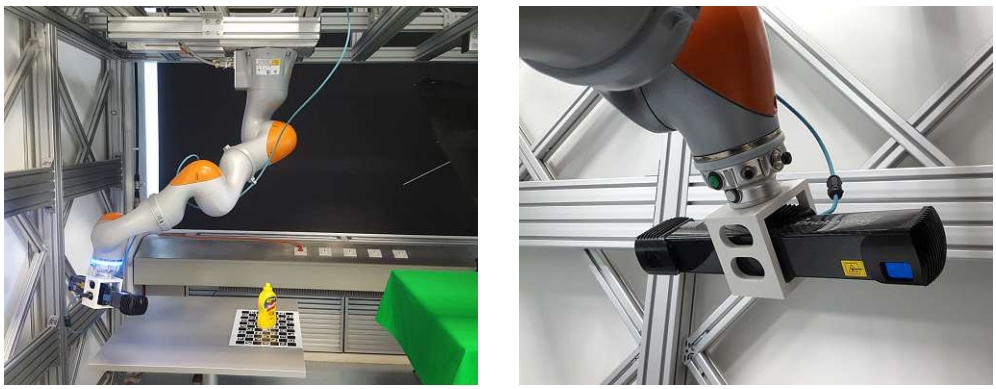
Figure 4.10: Selected iterations demonstrating the view planning algorithm for the power drill (■ infeasible views, ■ completed views, ■ planned views, ■ active view).

# 5 Experiments

In this chapter, experiments conducted to validate the developed reconstruction system are shown. First, the experimental setup is discussed and the process of robot-camera calibration is briefly explained. This calibration forms the basis of the results in the remainder of this chapter. Then, the alignment of scans with measured joint angles is compared to the alignment with a ChArUco board. Finally, the reconstruction of several objects with the non-model-based mode is presented. In particular, the proposed view planning algorithm is demonstrated and the quality of the scans is analyzed.

## 5.1 Experimental Setup

**Robot and Camera.** Unfortunately, the robotic system including the linear axes is not available at the time of writing. Thus, this work is validated in an experimental setup without linear axes. Figure 5.1a depicts this setup, where the base of the KUKA LBR iiwa 14 R820 is fixed to the ceiling. Hence, the total DoF is only seven instead of nine. To be able to scan objects from all sides, they are placed directly under the robot. Figure 5.1b shows how the PHOTONEO MotionCam-3D M is mounted on the robot's flange. This self-developed mounting was produced with a 3D printer.



(a) (b)

Figure 5.1: Experimental setup. (a) KUKA LBR iiwa 14 R820 without linear axes. The total DoF is seven instead of nine. (b) 3D-printed part for mounting the PHOTONEO MotionCam-3D M on the robot's flange.

**Marker Board.** A ChArUco board composed of $11 \times 7$ squares is used to estimate the camera pose for the following experiments. The squares of the ChArUco board are 35 mm

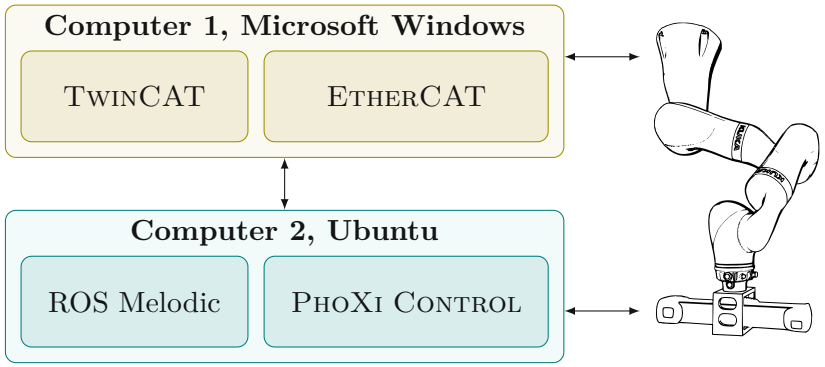



Figure 5.2: Communication setup for robot and depth camera. Computer 1 is used to control the robot. The reconstruction system is installed on computer 2, which is connected to the depth camera.

long and the ArUco markers within the board have a side length of 25 mm. To ensure a flat surface, the marker board is printed on an aluminum composite panel. Note that the robot-camera calibration explained below was done with another marker pattern to be able to apply Photoneo's calibration software.

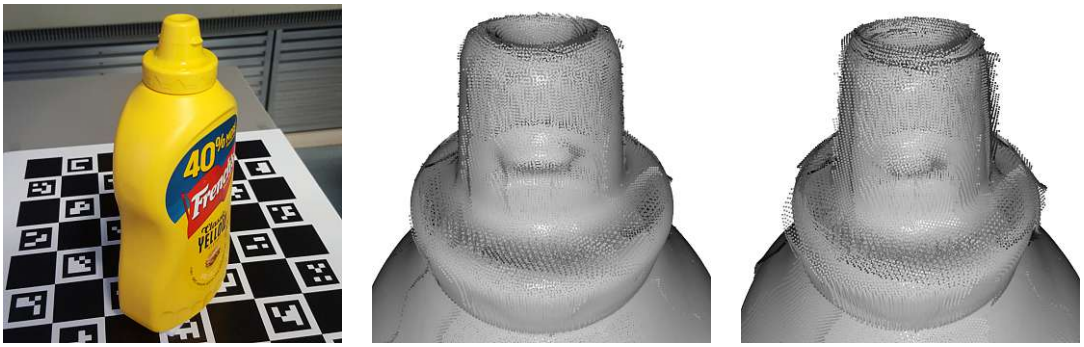
**Communication.** The system is operated with two computers, illustrated in Figure 5.2. On the first machine, the real-time automation software Beckhoff TwinCAT is installed, which is used to control the robot. The second computer executes the reconstruction software in a ROS environment. Commands for the robot are sent to computer 1 via TwinCAT's ADS protocol. In addition, Photoneo PhoXi Control is executed on computer 2 to communicate with the depth camera.

**Robot-Camera Calibration.** The homogeneous transformation

$$\mathbf{H}_{\mathcal{W}}^{\mathcal{C}} = \mathbf{H}_{\mathcal{W}}^{\mathcal{L}_9}\mathbf{H}_{\mathcal{L}_9}^{\mathcal{C}} \tag{5.1}$$

is of interest for two reasons. First, it is needed for motion planning to move the camera to desired views. Second, $\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}$ is required to align scans based on the measured joint angles when pose estimation with the marker board fails. Whereas $\mathbf{H}_{\mathcal{W}}^{\mathcal{L}_9}$ is known from CAD data of the robot, the transformation $\mathbf{H}_{\mathcal{L}_9}^{\mathcal{C}}$ is not known exactly. Thus, a robot-camera calibration was performed prior to the experiments.

Photoneo offers the Robot-Camera Calibration Tool for this purpose. A special marker pattern provided by Photoneo was printed in A4 format and placed in a static position on the table. Twelve scans were performed at arbitrary poses around Photoneo's marker pattern. For each scan, the end-effector's pose was determined by evaluating $\mathbf{H}_{\mathcal{W}}^{\mathcal{L}_9}(\tilde{\mathbf{q}})$ at the measured joint angles $\tilde{\mathbf{q}}$. Together with the estimated camera pose based on the marker board, the unknown transformation $\mathbf{H}_{\mathcal{L}_9}^{\mathcal{C}}$ was determined.



(a) Mustard bottle. (b) Alignment with markers. (c) Alignment with joint angles.

Figure 5.3: (a) Scanned mustard bottle with comparison of the alignment (b) based on pose estimation with the marker board and (c) based on measured joint angles.

## 5.2 Initial Alignment

The purpose of this section is to compare the alignment of scans based on pose estimation with the ChArUco board to the alignment based on the measured joint angles, see also the discussion in Section 2.4.2. This initial alignment is necessary to apply the local registration algorithm presented in Section 2.4.2.

To compare the alignment achieved with both methods, twelve scans of the mustard bottle depicted in Figure 5.3a were made. The camera was moved in steps of 30° around the object at a distance of about 0.6 m. Note that not only the first joint of the collaborative robot (corresponding to $q_3$ in Figure 2.7) was moved. Instead, a wide variety of robot configurations was chosen. This is important when validating the homogeneous transformation $\mathbf{H}_{\mathcal{W}}^{\mathcal{C}}$.

By estimating the camera pose based on the marker board, a tight initial alignment is achieved, see Figure 5.3b. In comparison, Figure 5.3c shows the same scans, but this time aligned based on the measured joint angles. Misalignments are clearly visible. However, considering that the robot is not calibrated, this is a remarkable performance. The advantage of this method is that no marker board is necessary. Nevertheless, for the following experiments, the marker board is prioritized and the alignment based on measured joint angles is used as a fallback when pose estimation fails.

## 5.3 View Planning

In this section, the proposed view planning algorithm of the non-model-based mode is demonstrated. For this purpose, three objects depicted in Figure 5.4 are scanned with distance parameter $d_c = 0.55\,\text{m}$. The motion planner assumes that the objects fit on the marker board to avoid collisions. Other assumptions regarding the geometric properties of the objects are not necessary.

The figures in the following subsections show the partially scanned object with an estimated bounding box. Furthermore, the overlaid ellipsoid with distributed views on its surface is illustrated. Again, the color scheme of Figure 2.5 is used to distinguish infeasible,

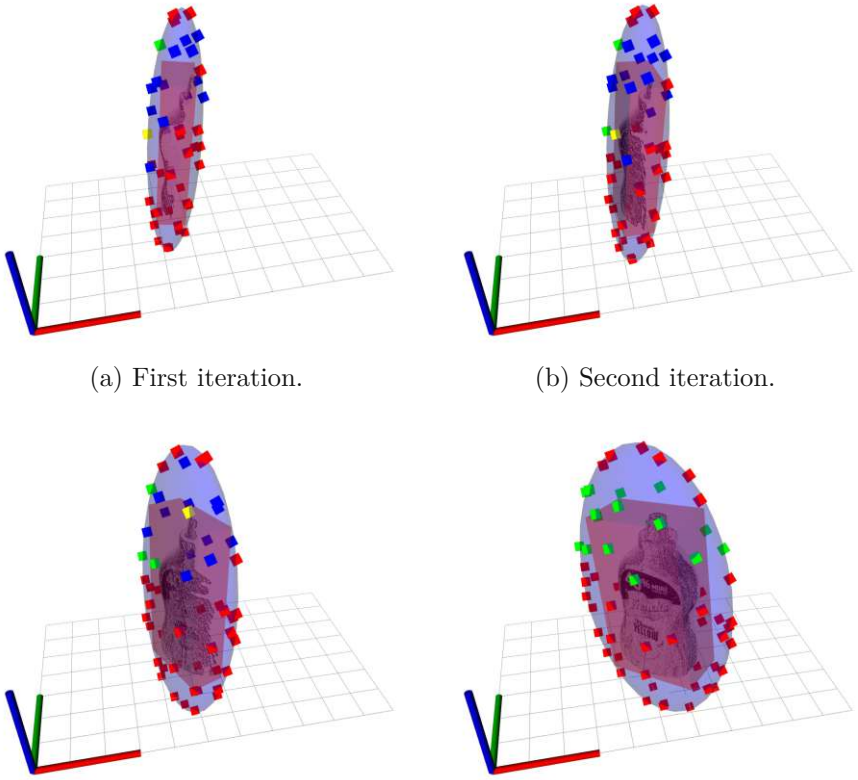
(a) Mustard. (b) Toy. (c) Wood block.

Figure 5.4: Objects selected to demonstrate the proposed view planning algorithm. The generated meshes are depicted in the second row.

completed, and planned views as well as the active view, i. e. they are respectively colored in red, green, blue, and yellow. In addition, the grid of the marker board is depicted for easier size comparison, together with the corresponding marker frame $\mathcal{M}$ (see also Figure 2.12).

### 5.3.1 Mustard Bottle

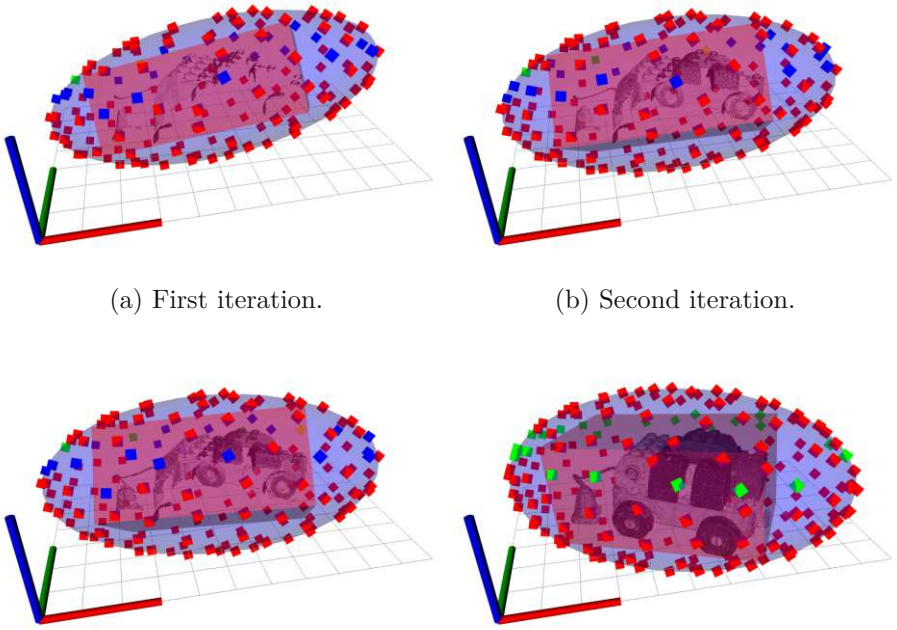
Figure 5.4a shows the first object, which is reconstructed with 16 scans. It is a mustard bottle from the YCB object and model dataset [37]. After the first iteration, depicted in Figure 5.5a, the estimated ellipsoid is still very narrow because only a small part of the object is captured. Infeasible views occur mainly on the lower part of the ellipsoid because here the camera would collide with the table. Therefore, the planned views are primarily found on the upper part of the ellipsoid. Furthermore, the object cannot be scanned directly from above because the object is placed below the robot, see Figure 5.1a. After the second iteration, illustrated in Figure 5.5b, a part of the mustard bottle's front side is already captured. It increases significantly after the third iteration, see Figure 5.5c. Finally, Figure 5.5d shows the scenario after the last iteration. The view planning algorithm was able to capture the unknown object while achieving a good distribution of views. This is also reflected in the computed mesh, which is illustrated in the second row of Figure 5.4a.



(a) First iteration. (b) Second iteration.

(c) Third iteration. (d) Last iteration.

Figure 5.5: Selected iterations demonstrating the view planning algorithm for the mustard bottle, see Figure 5.4a (■ infeasible views, ■ completed views, ■ planned views, ■ active view).



(a) First iteration. (b) Second iteration.

(c) Third iteration. (d) Last iteration.

Figure 5.6: Selected iterations demonstrating the view planning algorithm for the toy, see Figure 5.4b (■ infeasible views, ■ completed views, ■ planned views, ■ active view).

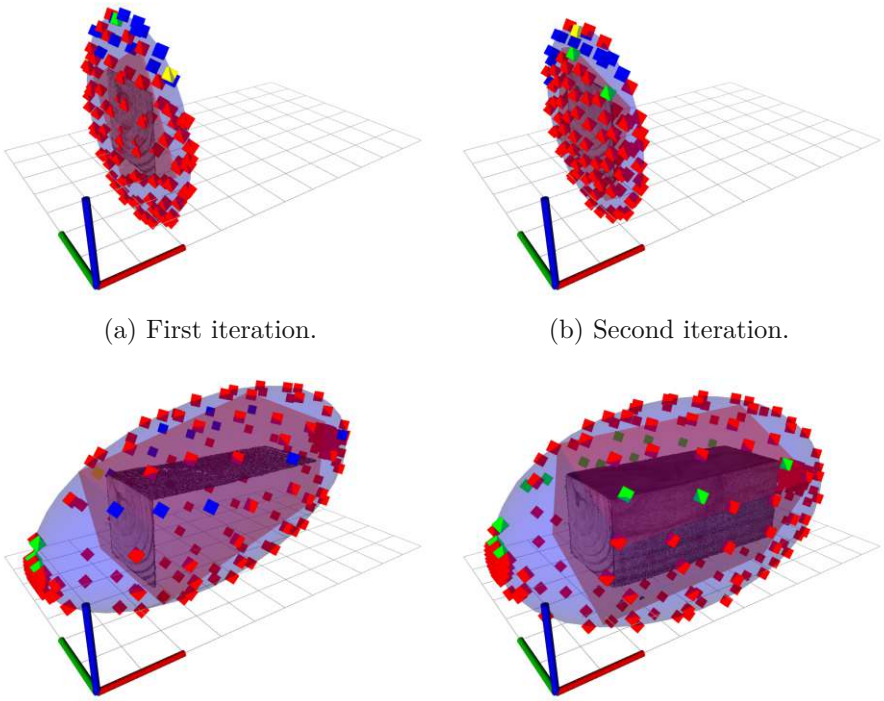
### 5.3.2 Toy

Next, the child's toy from Figure 5.4b is reconstructed with 24 scans. After the first iteration, a large part of the object is already captured. This is depicted in Figure 5.6a. The associated ellipsoid approximates the object very well. Unlike the mustard bottle, this object is flat, which results in many sampled views being above the object. However, since the object is placed directly below the robot, a lot of these views are infeasible. Ultimately, only a narrow strip remains for the planned views, which can also be seen in Figure 5.6b and Figure 5.6c. This is a result of the limited workspace of the robot because linear axes are not available. In the simulation with linear axes, see Section 4.4, such limitations were not observed. Nevertheless, the feasible views are evenly distributed around the object, as Figure 5.6d shows.

The generated mesh is depicted in Figure 5.4b. Especially at the bottom of the object and between the cyan and violet blocks, the mesh exhibits extrapolation effects. Apart from this, the rest of the object is reconstructed very well.

### 5.3.3 Wood Block

The last object in this section is the wood block depicted in Figure 5.4c, which is reconstructed with 16 scans. The motivation for scanning this object is to find out how the



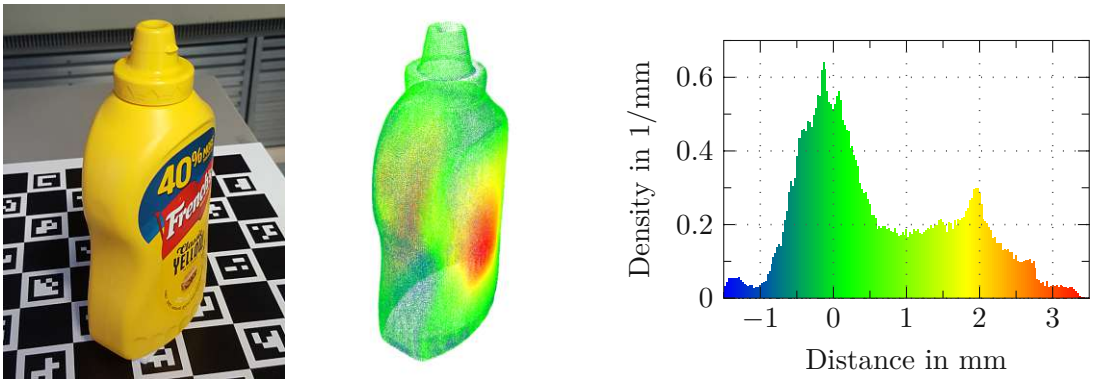
(a) First iteration. (b) Second iteration.

(c) Third iteration. (d) Last iteration.

Figure 5.7: Selected iterations demonstrating the view planning algorithm for the wood block, see Figure 5.4c (■ infeasible views, ■ completed views, ■ planned views, ■ active view).

view planning algorithm copes with an initial scan from a very unfavorable position. To do this, the wood block is scanned frontally from the face, so initially only a two-dimensional point cloud is visible.

Even after the first iteration, shown in Figure 5.7a, no new information about the depth of the object is available. After the second iteration, illustrated in Figure 5.7b, the ellipsoid minimally widens and a view at the top of the ellipsoid is selected for the next scan. As depicted in Figure 5.7c, this results in a scan capturing the top of the wood block. Accordingly, the computed ellipsoid captures most of the wood block and allows planning views from the sides. Figure 5.7d shows the last iteration where the wood block is already fully scanned. It can be seen that the bounding box estimated by Open3D is twisted with respect to the wood block. Hence, the corresponding ellipsoid is larger than necessary. However, the view planning algorithm was able to deal with the challenging initial scan.




(a) Physical object. (b) Point cloud. (c) Deviation to reference model.

Figure 5.8: Evaluation of the scanned mustard bottle.

## 5.4 Quality Analysis

This section is dedicated to the analysis of the accuracy that can be achieved with the developed reconstruction system. For this purpose, three objects are scanned and compared with their available 3D reference models. The comparison is performed by means of the program CloudCompare [65], an open-source project. Specifically, signed distances between the triangle mesh of the reference model and the point cloud obtained from the reconstruction system are calculated. This point cloud is the result of the locally registered and integrated scans before applying Poisson surface reconstruction, see Section 2.4.2 and Section 2.4.3 for details.

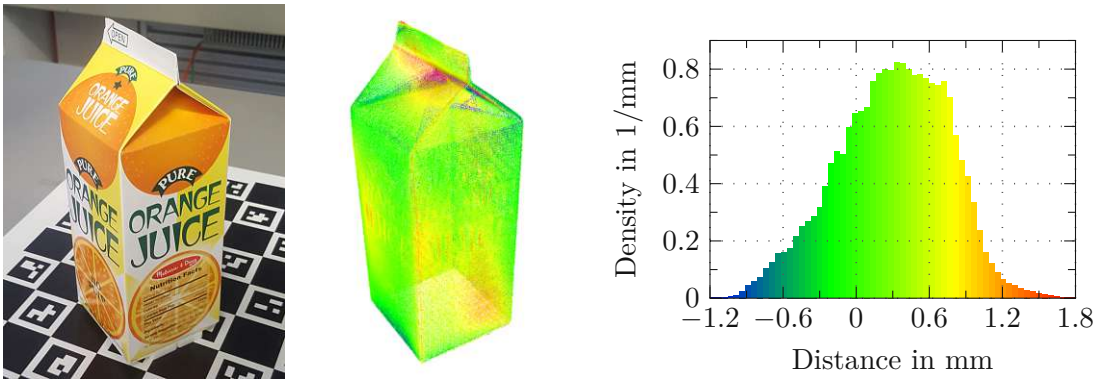
The following figures show respectively the physical object, the obtained point cloud, and a histogram of the calculated deviation to the reference model. The ordinate of the histogram is scaled so that the area under the plot is normalized. In order to establish a link between histogram and point cloud, both representations are consistently colored.

### 5.4.1 Mustard Bottle

As a first object, a mustard bottle from the YCB object and model dataset [37] is scanned and compared to its reference model. Figure 5.8 shows the physical object, the obtained point cloud, and the corresponding histogram of the deviations to the 3D model. The mean distance to the 3D model is 0.61 mm and the distribution has a standard deviation of 1.04 mm. It should be noted here that this benchmark object is nonrigid and may have deformed over time. Especially in the middle area of the mustard bottle, there are significant deviations of about 3 mm. Despite this broad distribution, approximately 51.6 % of the values lie in the interval ±0.6 mm, which is quite reasonable.

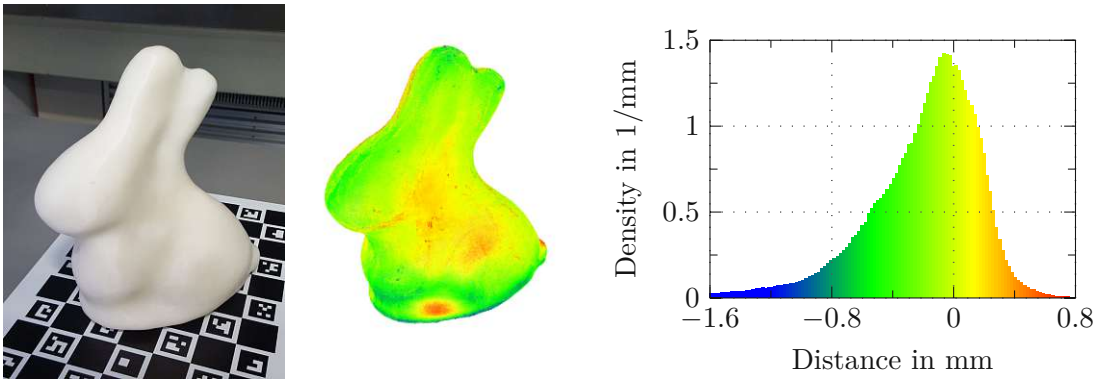
### 5.4.2 Orange Juice

The second object is an orange juice carton available in the Household Objects for Pose Estimation (HOPE) dataset [66] from Nvidia. It is also a nonrigid benchmark object. As Figure 5.9 shows, the distribution of deviations is much narrower than for the mustard




(a) Physical object. (b) Point cloud. (c) Deviation to reference model.

Figure 5.9: Evaluation of the scanned orange juice.




(a) Physical object. (b) Point cloud. (c) Deviation to reference model.

Figure 5.10: Evaluation of the scanned bunny.

bottle. The standard deviation is 0.63 mm and the mean is 0.26 mm. In this case, about 68.1 % of the values lie in the interval ±0.6 mm. Stronger deviations occur mainly at the bottom and in the area of the fold. About 96 % of the surface was captured.

### 5.4.3 Bunny

The previous two objects are deformable and therefore do not exactly match their reference model. Next, a professionally 3D-printed bunny is scanned and compared with its CAD model. Figure 5.10 depicts the corresponding evaluation. The deviation has a mean value of −0.2 mm and a standard deviation of 0.38 mm. Like for the orange juice, negative deviations occur primarily at the bottom of the bunny. About 49.4 % of the values lie in the interval ±0.2 mm. Taking the accuracy of 0.25 mm of the PHOTONEO MotionCam-3D M into account, this is a very good result. The obtained point cloud covers about 95 % of the bunny's surface. This is mostly due to the missing underside of the bunny's chin, which is hard to capture without placing the bunny on a pedestal.

Table 5.1 summarizes the mean and standard deviation for all of the discussed objects.

| Object | $\mu$ | $\sigma$ |
|---|---|---|
| Mustard Bottle | 0.61 mm | 1.04 mm |
| Orange Juice | 0.26 mm | 0.63 mm |
| Bunny | −0.20 mm | 0.38 mm |

Table 5.1: Summary of the deviation to the 3D reference model for all three scanned objects, expressed by mean $\mu$ and standard deviation $\sigma$.

The reconstruction of the bunny matches best to its reference model. Considering the fact that the bunny is also the most reliable object (because it is rigid), this highlights the remarkable precision achieved with the developed reconstruction system.

# 6 Conclusions and Outlook

This thesis dealt with the development of a robotic 3D reconstruction system. It was implemented in ROS, which makes it easily extensible. Simulation studies and experiments were conducted to validate this work. Several core components were treated, such as motion planning, sequence optimization, view planning, and object reconstruction. These components are now revisited and summarized.

## 6.1 Conclusions

The robotic system including the linear axes was implemented in MoveIt to utilize existing sampling-based motion planning algorithms. In a simulation study for a designed benchmark scenario, the RRT-connect algorithm was identified to be well suited for the given robotic system.

If multiple pose goals for scanning an object are known in advance, the possibility of sequence optimization arises. This was tackled as a Traveling Salesman Problem (TSP) and an algorithm proposed in the literature was implemented to solve it approximately. In a simulation, the performance was examined for a specific benchmark problem. The total duration of the trajectory was reduced with only little computation overhead. A further improvement was achieved by applying a heuristic to find good inverse kinematics solutions prior to the approximate solution of the TSP.

To scan unknown objects with minimal user interaction, the reconstruction system was extended by a novel view planning algorithm. It eliminates the need to specify views manually and only requires an initial scan to reconstruct objects autonomously. The proposed view planning algorithm is based on a geometric approximation of the object as an ellipsoid. This approximation is refined iteratively during execution. Views are distributed on the ellipsoid's surface by means of random sampling in combination with an artificial potential field. Concepts of sequence optimization were specialized to select the next best view. Experiments demonstrated that this view planning algorithm can cope even with very unfavorable initial scans. Difficulties in finding feasible robot configurations were observed in the experiments, which is due to the limited workspace of the 7-DoF robot and the necessary distance of the camera to the ellipsoid's surface.

For global registration of scans, a marker board was applied. As the experiments showed, a tight initial alignment can be achieved with this method. Furthermore, another approach based on the measured joint angles was tested in an experiment. With a robot-camera calibration, a satisfactory initial registration was obtained even though the robot itself was not calibrated. To refine the initial alignment of scans, an approach using a variant of the ICP algorithm in combination with pose graph optimization is used. A comparison of the resulting point cloud with a reference model was done for several objects. Especially for the 3D-printed bunny, a remarkable accuracy was achieved.

## 6.2 Outlook

Unfortunately, this work could not be validated on the whole robotic system with nine DoF. Therefore, the next step will be to test the reconstruction system in an experimental setup including the linear axes. Due to the increased workspace, this also results in improvements for the proposed view planning algorithm. Moreover, a height-adjustable table is available, which was not utilized for this work. A further step can therefore be to incorporate this additional DoF into the system as well.

With this extended system, it is to be analyzed how accurate the alignment of scans is based on the measured joint positions. It may be possible to dispense with the marker board altogether if high precision is not required. This would offer substantial advantages because the restriction regarding the visibility of markers is eliminated.

# Bibliography


[1] M. Costin, A. Ignat, O. Baltag, S. Bejinariu, C. Stefanescu, F. Rotaru, and D. Costandache, "3D breast shape reconstruction for a non-invasive early cancer diagnosis system," in *2nd International Workshop on Soft Computing Applications*, 2007, pp. 45–50.

[2] R. Levy and P. Dawson, "Reconstructing a thule whalebone house using 3D imaging," *IEEE MultiMedia*, vol. 13, no. 2, pp. 78–83, 2006.

[3] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *12th International Conference on Computer Vision (ICCV)*, IEEE, 2009, pp. 80–87.

[4] M. Bitzidou, D. Chrysostomou, and A. Gasteratos, "Multi-camera 3D object reconstruction for industrial automation," in *Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services*, Springer, 2013, pp. 526–533.

[5] L. Cruz, D. Lucio, and L. Velho, "Kinect and RGBD images: Challenges and applications," in *25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, IEEE, 2012, pp. 36–49.

[6] I. Makarov and D. Chernyshev, "Real-time 3D model reconstruction and mapping for fashion," in *43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020, pp. 133–138.

[7] R. A. Newcombe, A. Fitzgibbon, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, and S. Hodges, "KinectFusion: Real-time dense surface mapping and tracking," in *10th International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, 2011, pp. 127–136.

[8] R. Pito and R. Bajcsy, "A solution to the next best view problem for automated CAD model acquisition of free-form objects using range cameras," Department of Computer and Information Science, University of Pennsylvania, Tech. Rep., 1995.

[9] R. Pito, "A sensor-based solution to the "next best view" problem," in *13th International Conference on Pattern Recognition*, vol. 1, IEEE, 1996, pp. 941–945.

[10] M. Callieri, A. Fasano, G. Impoco, P. Cignoni, R. Scopigno, G. Parrini, and G. Biagini, "RoboScan: An automatic system for accurate and unattended 3D scanning," in *2nd International Symposium on 3D Data Processing, Visualization and Transmission*, IEEE, 2004, pp. 805–812.

[11] M. Karaszewski, R. Sitnik, and E. Bunsch, "On-line, collision-free positioning of a scanner during fully automated three-dimensional measurement of cultural heritage objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1205–1219, 2012.

[12] S. Larsson and J. A. P. Kjellander, "Motion control and data capturing for laser scanning with an industrial robot," *Robotics and Autonomous Systems*, vol. 54, no. 6, pp. 453–460, 2006.

[13] ——, "Path planning for laser scanning with an industrial robot," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 615–624, 2008.

[14] S. Khalfaoui, R. Seulin, Y. Fougerolle, and D. Fofi, "An efficient method for fully automatic 3D digitization of unknown objects," *Computers in Industry*, vol. 64, no. 9, pp. 1152–1160, 2013.

[15] H. Kwon, M. Na, and J.-B. Song, "Rescan strategy for time efficient view and path planning in automated inspection system," *International Journal of Precision Engineering and Manufacturing*, vol. 20, pp. 1747–1756, 2019.

[16] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM computing surveys*, vol. 35, no. 1, pp. 64–96, 2003.

[17] W. R. Scott, "Model-based view planning," *Machine Vision and Applications*, vol. 20, no. 1, pp. 47–69, 2009.

[18] M. D. Kaba, M. G. Uzunbas, and S. N. Lim, "A reinforcement learning approach to the view planning problem," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 5094–5102.

[19] D. Eberly, *Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid*, Geometric Tools, 2013. [Online]. Available: `https://www.geometrictools.com` [Accessed Oct. 28, 2021].

[20] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, 1983.

[21] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[22] B. Siciliano, *Robotics: Modelling, Planning and Control*, ser. Advanced textbooks in control and signal processing. London: Springer, 2010.

[23] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *International Conference on Robotics and Automation (ICRA)*, vol. 2, IEEE, 1985, pp. 500–505.

[24] R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 224–233, 1985.

[25] J.-C. Latombe, *Robot Motion Planning*, ser. The Kluwer international series in engineering and computer science. Boston: Kluwer, 1991.

[26] C. ó'Dúnlaing, M. Sharir, and C. Yap, "Retraction: A new approach to motion-planning," in *15th annual ACM symposium on theory of computing*, ACM, 1983, pp. 207–220.

[27] C. ó'Dúnlaing and C. K. Yap, "A "retraction" method for planning the motion of a disc," *Journal of Algorithms*, vol. 6, no. 1, pp. 104–111, 1985.

[28] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[29] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[30] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[31] M. G. Calkin, *Lagrangian and Hamiltonian Mechanics*. Singapore: World Scientific, 1996.

[32] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York: Wiley, 1989.

[33] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 2, pp. 279–298, 2015.

[34] D. L. Applegate, *The Traveling Salesman Problem: A Computational Study*, ser. Applied Mathematics. Princeton: Princeton University Press, 2006.

[35] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning tours of robotic arms among partitioned goals," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 207–223, 2006.

[36] M. Saha, G. Sánchez-Ante, and J.-C. Latombe, "Planning multi-goal tours for robot arms," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2003, pp. 3797–3803.

[37] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[38] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 143–152.

[39] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial markers for pose estimation: Overview, applications and experimental comparison of the ARTag, AprilTag, ArUco and STag markers," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 4, pp. 1–26, 2021.

[40] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[41] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120–125, 2000.

[42] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[43] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 1991, pp. 2724–2729.

[44] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3-D Digital Imaging and Modeling*, IEEE, 2001, pp. 145–152.

[45] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 5556–5565.

[46] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," *Computer Graphics Forum*, vol. 36, no. 1, pp. 301–329, 2017.

[47] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Fourth Eurographics symposium on geometry processing*, Eurographics Association, 2006, pp. 61–70.

[48] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[49] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, IEEE, 2009, pp. 1–6.

[50] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, Aug. 2010. [Online]. Available: `http://www.programmingvision.com/rosen_diankov_thesis.pdf` [Accessed Aug. 6, 2021].

[51] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[52] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!" *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[53] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A MoveIt! case study," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3–16, 2014.

[54] I. Sucan and J. Kay, *URDF: Unified Robot Description Format*. [Online]. Available: `http://wiki.ros.org/urdf` [Accessed Sep. 7, 2021].

[55] I. Sucan, *SRDF: Semantic Robot Description Format*. [Online]. Available: `http://wiki.ros.org/srdf` [Accessed Sep. 7, 2021].

[56] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2000, pp. 995–1001.

[57] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Robotics: Science and Systems*, vol. 8, MIT Press, 2013, pp. 209–216.

[58] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, IEEE, 2004, pp. 2149–2154.

[59] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[60] D. Hershberger, D. Gossow, J. Faust, and W. Woodall, *3D visualization tool for ROS*. [Online]. Available: `http://wiki.ros.org/rviz` [Accessed Dec. 3, 2021].

[61] S. Glaser, W. Woodall, and R. Haschke, *Xacro: XML macro language*. [Online]. Available: `http://wiki.ros.org/xacro` [Accessed Sep. 7, 2021].

[62] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[63] D. Hsu and J.-C. L. R. Motwani, "Path planning in expansive configuration spaces," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 1997, pp. 2719–2726.

[64] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, Oct. 1998.

[65] D. Girardeau-Montaut, *CloudCompare*. [Online]. Available: `http://cloudcompare.org` [Accessed Nov. 24, 2021].

[66] *Household objects for pose estimation (HOPE)*, Nvidia Corporation. [Online]. Available: `https://github.com/swtyree/hope-dataset` [Accessed Nov. 23, 2021].

# Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, Jänner 2022

Martin Fraunhofer