TUM School of Computation, Information and Technology
Technische Universität München

X-ray Computed Tomography with a Robotic Sample Holder

Erdal Pekel

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr. Cristina Piazza

Prüfer der Dissertation:
1. Priv.-Doz. Dr. Tobias Lasser
2. Prof. Dr. Franz Pfeiffer

Die Dissertation wurde am 26.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 17.10.2023 angenommen.

Abstract

X-ray computed tomography is used in science, medicine, and other fields to obtain three-dimensional absorption information about an object with a transmission-based imaging modality. A three-dimensional image is obtained using multiple two-dimensional absorption images of the object under study with known geometries, also called image reconstruction in a process called computed tomography.

One key parameter of image reconstruction quality is the diversity of the geometries at which the objects are exposed to the X-rays. The limitations of static sample mounting mechanisms can be overcome by introducing flexible actuators. For example, robotic c-arms for patient scans allow the acquisition of images from more diverse geometries in medical applications.

Industrial robotic arms offer a more comprehensive range of movements than conventional mechanisms. They have become more affordable for a broader audience with the mass production of electrical and mechanical components. At the same time, their control and software integration has become more feasible with the emergence of open-source software initiatives that aim to standardize and simplify the use of such robotic arms.

In this thesis, we explore using a robotic arm as a sample holder for X-ray computed tomography for its superior flexibility without sacrificing image quality. The methods and algorithms introduced herein tackle the challenges of this novel holding mechanism.

In the first part of this thesis, we develop a calibration procedure that tackles the most fundamental limitation of the robotic arm: repeated placement accuracy. We introduce an intermediate sample holder item design connected to the robotic arm acting as a calibration target.

In the second part, we analyze and improve the system's ability to execute more advanced trajectories. We maximize the robotic arm's reach with new designs of the previously introduced sample holder item. The advantage of advanced geometries, such as spherical trajectories over circular geometries in image quality, is demonstrated.

In the third part, we develop routines and algorithms that reduce acquisition times and radiation dose by generating optimized acquisition geometries for the given sample without prior knowledge. We introduce a more advanced, runtime-optimized acquisition strategy, where we analyze the acquired absorption images while the robotic arm executes the trajectory. We modify the current trajectory based on the analysis of these images.

The proposed system is a step towards higher image reconstruction quality in X-ray CT systems. The system allows flexible sample placement with advanced acquisition trajectories. Additionally, it reduces acquisition times and radiation dose exposure with sample-specific trajectories.

Zusammenfassung

Die Röntgen Computertomographie wird in der Wissenschaft, Medizin und anderen Anwendungsgebieten für die Berechnung des drei-dimensionalen Absorptionsmodells eines Messobjekts (Probe) mittels einer transmissions-basierten Bildgebungsmodalität verwendet. Das drei-dimensionale Absorptionsmodell wird aus einer Reihe von zwei-dimensionalen Aufnahmen der untersuchten Probe aus verschiedenen Aufnahmewinkeln ermittelt; Diesen Prozess der Rekonstruktion bezeichnet man als Computertomographie.

Ein entscheidender Parameter für die Qualität der resultierenden Rekonstruktion ist die Zusammensetzung der Aufnahmen: Die Bildqualität wird durch die vielfältige Auswahl von Aufnahmewinkeln sichtbar verbessert. Die Beschränkungen statischer Mechanismen für die Befestigung von Proben können durch den Einsatz flexibler Aktuatoren überwunden werden. Der Einsatz robotischer C-Arme für die Patientenuntersuchung in der Medizin ermöglicht beispielsweise Aufnahmen aus einer großen Anzahl unterschiedlicher Winkel.

Industrielle Roboterarme zeichnen sich im Vergleich zu konventionellen Bewegungsmechanismen durch eine umfassendere Bewegungsfreiheit aus. Ferner wurde die Anschaffung von Roboterarmen durch die industrielle Produktion elektrischer und mechanischer Komponenten erschwinglicher für die breite Mehrheit. Zugleich wurde ihre Softwareintegration und Regelung praktikabler durch neue Initiativen für freie Software in diesem Bereich. Diese Initiativen beabsichtigen den einfachen und standardisierten Einsatz der eben genannten Roboterarme.

In dieser Arbeit setzen wir uns mit der Nutzung eines Roboterarms als Probenhalter für Röntgen Computertomographie auseinander. Die Flexibilität des Roboterarms soll genutzt werden, ohne die Bildqualität zu beeinträchtigen. Die Algorithmen und Methoden in dieser Arbeit lösen die Probleme und Herausforderungen dieses neuartigen Probenhalters.

Im ersten Teil dieser Arbeit entwickeln wir einen Kalibrierungsalgorithmus, der ein fundamentales Problem des robotischen Probenhalters löst: Die wiederholte, ungenaue Platzierung von Proben durch den Roboterarm. Für diesen Algorithmus führen

wir ein Zwischenstück als Kalibrierungsziel zwischen dem Greifer des Roboterarms und der Probe ein.

Im zweiten Teil dieser Arbeit analysieren und verbessern wir die Ausführung fortgeschrittener Trajektorien mit dem robotischen Probenhalter. Wir optimieren die Fähigkeit des Roboterarms, einen Großteil der möglichen Rotationen im dreidimensionalen Raum auszuführen. Hierfür führen wir verschiedene Prototypen und Formen des vorhin erwähnten Zwischenstücks ein, das als Kalibrierungsziel dient. Wir legen den Vorteil von fortgeschrittenen Trajektorien im Vergleich zu Kreistrajektorien dar und untermauern dies mit den erzielten Messergebnissen und Unterschieden in deren Bildqualität.

Im dritten und letzten Teil entwicklen wir einen Algorithmus zur Reduzierung der Aufnahmezeit und zur Minimierung der Strahlendosis bei der Nutzung des robotischen Probenhalters. Unser System erreicht dieses Ziel durch den Einsatz optimierter Aufnahmetrajektorien. Wir führen in diesem Teil der Arbeit eine Methode ein, die noch während der Probenmessung einzelne Aufnahmen analysiert und die zukünftige Trajektorie der Aufnahme basierend auf dieser Analyse anpasst. Unser System nimmt die Anpassung der Trajektorie ohne Vorwissen über die Probe vor.

Unser System trägt zur Verbesserung der Bildqualität in Systemen der Röntgen Computertomographie bei. Das System ermöglicht die flexible Platzierung von Proben mit fortgeschrittenen Aufnahme-Trajektorien. Zudem verkürzt es Aufnahmezeiten und mindert die Strahlendosis durch den Einsatz von optimierten Trajektorien.

Acknowledgements

In the last four and a half years, I have been honored to work on this exciting project: the robotic sample holder for X-ray computed tomography. With my supervisors, we made surprising discoveries and researched a broad and diverse set of problems during these years. I want to thank those that contributed to my research and ultimately to my successful Ph.D. with their invaluable support.

First, I would like to thank my supervisor Tobias Lasser who trusted me with this project very early on, beginning with my master's thesis. He was always there when I needed his guidance, and in all these years, he never made me feel that I could not execute this project and solve the problems that arose. He gave me the freedom that I needed to try out various approaches before settling on the ideal solution for a problem; I am grateful for the opportunity to work with and learn from him. Thank you very much!

Second, the awesome CIIP research group with a team that always supported each other. We were always excited to talk to each other when we could sit together and exchange ideas. The team members Alessandro, Josue, Theo, Jonas, David, and Anca were enjoyable to be around; I always enjoyed talking to them, despite my busy schedule as a Ph.D. student. The one thing that I regret is that we didn't meet more often, but that is almost certainly the reality in any passionate team of researchers.

Third, I would like to thank the people from the E17 research group at the physics department. Beginning with Prof. Franz Pfeiffer, who supported my project throughout my Ph.D. by encouraging excellent researchers from his group to help me with my project when I needed them. Prof. Pfeiffer was also immensely generous with providing laboratory space for experiments and other types of equipment like a 3D printer or access to a very professional workshop for prototyping mechanical parts. I would also like to thank Dr. Martin Dierolf for his support, beginning with the first day of my Ph.D. Martin was always polite and easy to approach, and he spent numerous hours and days with me to support my experiments in the laboratory. I also want to thank Niko Gustschin, who has 3D printed 10 or 20 different mechanical parts for my thesis without complaining about the frequency at which I requested new parts. Last but not least, I want to thank Dr. Florian Schaff, who helped me

immediately upon request with an experiment in the laboratory just before I was getting ready to start writing my first journal manuscript. And in general, I thank the numerous Ph.D. students and researchers at E17, whom I forgot to mention, for their support.

In the closing paragraph, I thank my friends and family for their **immense** support throughout these years. Pursuing a Ph.D. is comparable to walking in a labyrinth full of twists and turns. And my friends and family felt every single one of these bumpy turns; I am confident that they did the Ph.D. with me and have suffered almost as much as I did. These people are still very important to me, and I cannot express my gratitude for their endless support.

Contents

Part I

Introduction

1 Introduction

1.1 Motivation

Robotic manipulators are getting increasingly popular in tomographic imaging setups. The industrial, off-the-shelve robotic arms market is very competitive, and manufacturers offer a wide range of configurations for arms. The customer can decide on the overall size of the robotic arm, its maximum reach, degrees of freedom, gripper type, and other criteria. On the other hand, custom robotic arms are more expensive than off-the-shelve parts, and it takes a significant amount of research time and other customer and use-case-specific considerations until they reach the market.

Some critical considerations exist for tomographic systems and X-ray computed tomographic (CT) systems specifically. For example, the size and type of the sample to be measured will influence the typical radiation dose, allowed measurement time, and physical size of the CT setup. For medical systems, the radiation dose is a critical measure, and the physical setup and measurement process need to account for the dose limits that ensure a healthy experience for the patient. The measurement time is also critical in a medical setting, as the patient becomes uncomfortable over time. The physical size of the setup largely depends on the sample size and the type of source that produces the rays. The sample type is also a critical consideration as undesired material compositions and sample structure can lead to effects like beam-hardening on the measurement images and poor reconstruction image quality.

Robotic manipulators offer numerous benefits for tomographic systems. The manipulator's flexibility enables more complex trajectories, e.g., spherical, helical, and sample-specific. Additionally, advanced mounting mechanisms and gripper types can reduce occlusions with the sample. Robotic arms offer these benefits without imposing restrictions on the sample's size and weight, as with the correct configuration, the arm can move samples of various sizes and shapes.

The introduction of robotic manipulators in tomographic imaging systems can also lead to difficulties in the measurement process. The repeated placement accuracy of robotic arms increases with increasing flexibility. With every added degree of

freedom, the number of potential configurations for the robotic arm to reach a specific goal increases. At the same time, the accumulated error of the individual joints also increases. The problem we described here necessitates the introduction of geometric calibration algorithms for identifying the exact position and orientation of the sample for tomographic imaging purposes.

There are numerous ways to introduce robotic manipulators in tomographic imaging systems. C-arms are one way to add flexibility to patient scans in a medical environment. For industrial robotic arms specifically, two types of setups are possible. The robotic arm can move the sample between the statically mounted source and detector (see Fig. 1.1 (a)). This type of setup is economically more feasible than using two arms since only a single robotic arm is needed. However, it is more prone to occlusions on the measurements since individual arm links can interfere with the X-ray beam. Moreover, it doesn't allow non-rigid samples as the sample's internal structure would change with the arm's movements. Another possibility is to move the source and detector with one robotic arm each and mount the sample statically between them (see Fig. 1.1 (b)). This setup is more expensive since two robotics arms are needed instead of one, but it is required for heavy samples or samples that cannot be moved in the given setup because of the restricted room or other reasons. Additionally, the setup with two arms enables measurements for moving samples (e.g., a sample sitting in liquid). The internal structure of moving samples changes with the single-arm setup as the sample moves with the arm. With the two-arm setup, the sample is mounted statically and does not move.

This work integrates a flexible robotic arm with seven degrees of freedom as a sample holder within a laboratory X-ray Computed Tomography (CT) setup with a suitable calibration mechanism. The arm adds flexibility to the setup as a sample holder by enabling arbitrary rotation and placement of the sample. The subjects of this work are the methods and algorithms required to execute various trajectories safely. Integrating a robotic arm in X-ray CT systems is a step towards improved image quality, reduced radiation dose and scan times, and more flexible acquisition schemes.

The safe execution of advanced trajectories with the robotic arm for lab-sized samples is the primary subject of this thesis. Advanced trajectories include circular, spherical, and optimized, sample-specific trajectories. With sample-specific trajectories, we offer an operating mode that optimizes the acquisition trajectory depending on the structure and absorption characteristics of the given sample. We aim to improve reconstruction image quality for complex samples and reduce scan times.

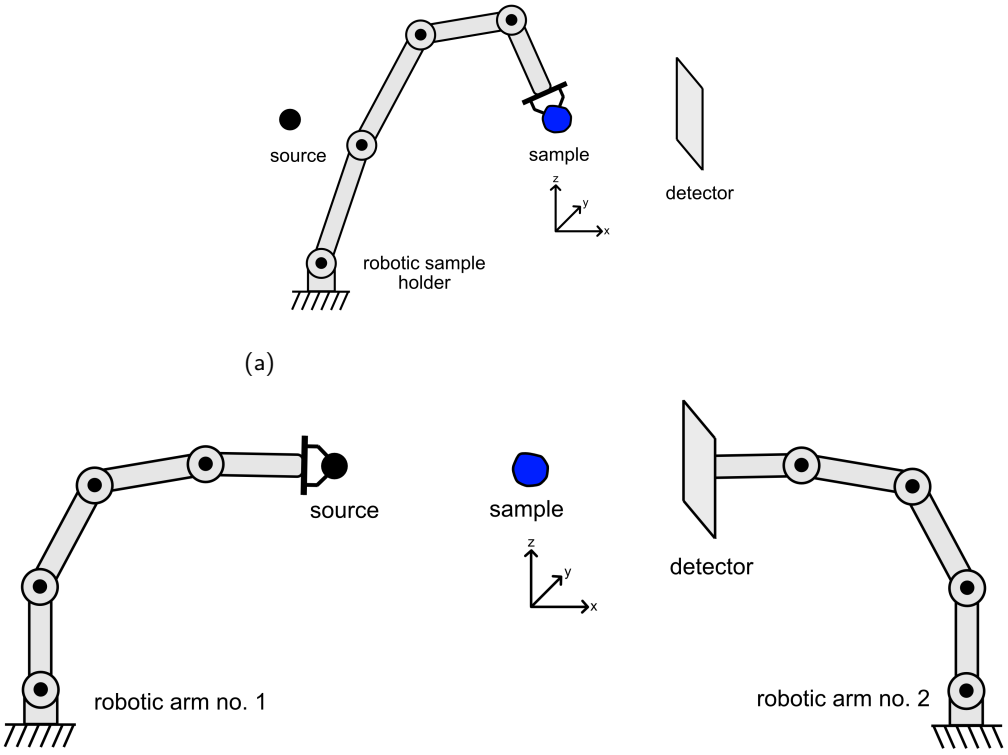


Fig. 1.1.: Robotic computed tomography setups. We illustrate two different kinds of robotic computed tomography setups. In (a), the robotic arm acts as a sample holder by holding and manipulating the sample. In (b), two robotic arms move around the sample, one holding and manipulating the source and the other moving the detector.

1.2 State of the Art

Robotic arms were also used in the past in computed tomography systems [Zie+20; LHH19; @Hea21; Her+21]. The main difference to our work is the kind of robotic arm used. It offers a higher flexibility than the robotic arms used in related work due to its seven degrees of freedom and two fingers, making chained pick-and-place tasks possible without user intervention.

Integrating a robotic arm with high degrees of freedom is a challenging task, as with every added degree of freedom, the placement accuracy of the arm at the end-effector decreases. The electrical motors at the arm's joints operate within a specified accuracy range. The placement error at each joint adds up to 0.1 mm in total with the robotic arm that we use [@EMI20]. The placement error increases on the detector image due to the magnification effect with projective imaging geometry.

The insufficient placement accuracy of the robotic arm can be fixed with a suitable calibration mechanism. Many methods exist in the literature for the general geometric calibration problem [LZL10; Cho+05; Rob+09]. In our work, we introduce a new method, as existing approaches do not apply to a robotic arm like ours or are limited to specific acquisition geometries.

1.3 Thesis Outline

In this thesis, we succeed in three steps.

In the first step, we integrate the robotic arm into an existing laboratory X-ray CT environment for the execution of conventional circular trajectories. At the end of this stage, we aim to match the reconstruction image quality of a conventional, electrical rotating stage. We design and manufacture a purpose-built sample holder part with an embedded geometric structure to calibrate the position and orientation of the sample for later use in the reconstruction step. We also implement a collision detection mechanism to prevent the arm from colliding with the equipment in its environment.

In the second step, we optimize our software implementation and algorithms for the seamless execution of spherical trajectories. We show the system's ability to cover a high percentage of all possible sample rotations on the sphere with a suitable sample holder part design and fine-tuning of the motion planning pipeline. We execute spherical trajectories without causing occlusions of the sample with the robotic arm's links on the detector images.

In the third and final step, we optimize the spherical acquisition scheme from step two for the given sample when no information is provided about it in advance. We optimize the trajectory at run-time without introducing additional measurement time compared to conventional spherical trajectories.

2 Fundamentals of Industrial Robotics

This chapter will introduce the most fundamental concepts of industrial robotics. We will begin with the configuration space, the underlying concept for solving the motion planning problem. Next, we will discuss how the motion planning problem is solved based on the configuration space. In the last two sections, we provide a brief outline of the forward and inverse kinematics problems essential to the robot's interaction with its operating environment.

2.1 Configuration Space

The content of this section is based on the corresponding chapters in [LaV06].

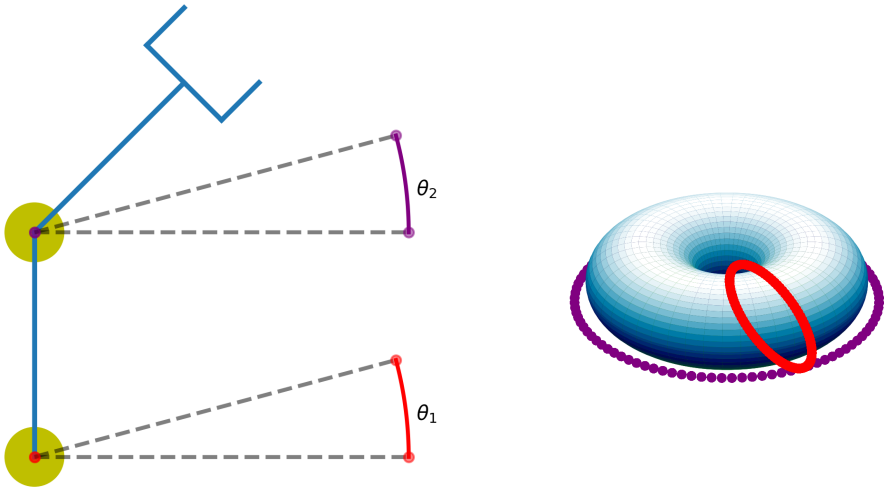
In robotic applications, one of the fundamental concepts is the configuration space. It provides a mathematical tool for the motion planning problem responsible for calculating feasible and safe paths for the robot. This section's mathematical concepts and assumptions are fundamental to solving the motion planning problem.

The configuration space is constructed by considering all possible transformations from the robot's state space. The types of joints of the robot define the state space. An example is displayed in figure 2.1. The *2R* robotic arm in this figure has two rotational joints in an open-loop kinematic chain.

2.1.1 Task-space and Work-space

Two important concepts that come up very often when operating a robot are the task-space and the workspace.

The *task-space* defines the space in which the robot operates naturally. This space is defined independently from the robot and solely dependent on the operator's perspective. In Euclidean space, the operator could point at any spot in his operating



(a) *2R* Robotic arm with two rotational joints (b) Torus: $T = S^1 \times S^1$

Fig. 2.1.: Configuration space. We illustrate the configuration space concept with a robotic arm and a visualization of the corresponding space. In (a), a robotic arm with two rotational joints is illustrated. In (b), the configuration space (Torus) of the robotic arm in (a) is visualized.

room and define the set of all such points as task-space without ever relating to a robot.

The *workspace* defines the configurations the robot's end-effector can reach. It is typically defined and constrained by the robot's number and type of joints. The physical properties of the robot are fixed in its design process. For example, a proper choice for the workspace of a seven degrees-of-freedom (DoF) industrial robotic arm is all positions and orientations that the tip of the end-effector can reach.

There is a clear distinction between the task-space, workspace, and configuration space: A specific point in the task or workspace might be reachable by multiple robot joint configurations; this is easily possible with high DoF robotic arms, as they are more flexible and hence can provide multiple distinct configurations of their joints to reach the same goal.

2.1.2 Topology

The configuration space's topology maps the robotic arm's physical properties to a mathematical space with well-defined operations. The notion of topology is essential for defining a smooth mapping between the task- and workspace of the robot.

The properties of the robot's joints determine the topology. There are six types of joints: rotational, prismatic, screw, cylindrical, spherical, and planar. This work focuses on rotational and prismatic joints, the most common types of joints in industrial robotic arms. For rotational joints, a range of angle values specifies the configuration; for prismatic joints, a range of real values specifies the maximum distance the joint can move.

The configuration space is modeled as a *manifold* to behave like a surface at any configuration point. Manifolds are general enough to represent the transformations that result from all joint types mentioned above while providing nice mathematical properties for the motion planning step that we will discuss later. The cartesian product of the individual manifolds of the joints represents the combination of a chain of links connected by joints. For rotational joints, the possible configurations lie on the 1d manifold, which is the unit circle defined by

$$\mathbb{S}^1 = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 = 1\}. \tag{2.1}$$

For every x and y, the circle defines a rotation angle θ. The manifold that describes the possible transformations for prismatic joints is $\mathbb{R}$.

The last concept we will introduce before defining the manifold of our example robotic arm is manifold *identification*. With manifold identification, we can reduce the dimensions of a manifold for better visualization and more straightforward motion planning in later steps. For example, the unit circle $\mathbb{S}^1$ introduced in eq. 2.1 can be simplified to the value range $[0, 2\pi]$ where the start and end values of the closed set are set to be identical. This identification simplifies the configuration space topology as it reduces to a small value range, where values exceeding 2π are mapped back to continue at the starting point of the identification point, 0.

For the *2R* robotic arm in Fig. 2.1a with two rotational joints, the configuration space can be defined as a *Torus*:

$$\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1. \tag{2.2}$$

The Torus in eq. 2.2 is defined by the cartesian product of two unit circle manifolds with identifications at 0 and 2π, which restricts the configuration space to the cartesian product of the two sets $[0, 2\pi]$ and reduces the size of the manifold compared to the cartesian product of two unit circles defined by $\mathbb{S}^1 \times \mathbb{S}^1$. For $\mathbb{S}$, angles exceeding 2π increase the area of the manifold.

2.1.3 Obstacles

An important observation is that obstacles must be added to the configuration space. Obstacle regions are necessary for two reasons: collisions of the robot's links with each other are possible, and external objects might be located inside the robot's reachable region. Obstacles are encoded inside the configuration space with approximations of their actual shapes. While different methods for modeling the obstacles in the configuration space exist, it is essential to understand that this task is not trivial. Different types of obstacles cause varying deformations in the configuration space when in motion. Additionally, the topology of the configuration space needs to be factored in, as different topological spaces allow for different deformations of bodies and obstacles.

While the different combinations of obstacle types and their modeling in different topological spaces are outside this work's scope, we will introduce critical mathematical notation that will be used in the upcoming sections for configuration space obstacles. Let C be the set of all possible configurations of the robot and let W be the workspace, where $O \subset W$ are the set of obstacles contained therein and $A \subset W$ is the robot. The *obstacle region* is defined as all configurations of the robot A that intersect with the obstacles O in the task-space W. The configurations that do not fall into this space are called *free space*: $C_{free} = C \setminus C_{obs}$.

2.2 Motion Planning

The content of this section is based on the corresponding chapters in [LaV06].

The critical idea of motion planning is determining if a robot configuration is reachable from another configuration. Two configurations are connected by a continuous *path*. A path on a topological space M is defined as $\tau : [0, 1] \mapsto M$, which ensures that all elements on the path are in the configuration space. If the topological space M is a manifold, we can assume that it is *path connected*, hence that there exists a path between two configurations $m_1, m_2 \in M$.

While there are different types of connectedness, each restricting the types of paths that are allowed for connecting two configurations, we will restrict ourselves to *simply connected* paths. Enforcing paths to be simply connected ensures that two different paths connecting the same start and end configurations can be *warped* into one another continuously, meaning that no "jumps" occur during this process. This property is also called *homotopy*. The assumption of simply connected paths on

our topological space results in nice properties for the control and motion planning process in later steps.

There are many approaches to solving the motion planning problem, but we will treat sampling-based planning in this work. Sampling-based planning is the most widely adopted approach, as it does not require explicitly constructing obstacles in the configuration space. Instead, it checks after sampling a configuration if it is inside the obstacle space before connecting it to a path. The omission of obstacle construction is a considerable advantage in algorithm implementation and complexity. However, it cannot provide the same guarantees for finding a solution as the combinatorial planning approach, which we will not treat in this thesis.

2.2.1 Sampling-based Motion Planning

Sampling-based motion planning is performed in three steps. In the first step, a suitable algorithm samples points in the configuration space for a fixed amount of time. Secondly, the points are connected to each other without passing obstacles to create a graph. Finally, the desired start and end configurations are connected with each other if there is a path on the graph. While there are slight variations to step two, the general scheme of the algorithm remains similar. In the following, we will discuss the key concepts for each phase.

The configuration space should be sampled based on different criteria in the sampling step. A critical criterion is the denseness of the sampling sequence. It reflects the desirable property that the sampled sequence of points gets arbitrarily close to every element in the space over which it samples. With *uniform random sampling*, the denseness of the sample depends on the quality of the pseudo-random number generator. Pseudo-random number generation introduces deterministic behavior that hinders uniform sampling over the configuration space. An alternative to random sampling is *low-dispersion sampling*. Here, the sampling algorithm chooses samples to minimize the largest empty area in the configuration space. When sampled on a grid, the samples align with the coordinate system axis. An alternative sampling scheme exists if the alignment of the samples with the coordinate axis is not desired: *low-discrepancy sampling*. With this technique, the samples are sampled uniformly over the configuration space while avoiding alignments of the samples. An appropriate measure for determining the discrepancy of a set of samples is to check the area they cover compared to the entirety of the configuration space. If the covered area ratio aligns with the sample ratio over all possible areas, the discrepancy is considered desirable.

The second and third steps of sampling-based motion planning are discussed in detail in the following sections.

2.2.2 Collision Detection

We have stated earlier that sampling-based motion planning depends on a separate collision detection algorithm. The sampled points need to be checked individually for collisions with the collision space C_{obs}. We will cover the hierarchical collision detection algorithm as an example.

In *hierarchical collision detection*, the body checked for the collision is split into smaller bodies with a tree structure consisting of groups of individual vertices, determining dependencies of parts with each other. In the next step, the vertices are grouped into bounding regions approximated by appropriate geometrical shapes. Increasing the number of splits in the tree also increases the number of geometric shapes and the computational complexity. Examples of suitable geometric bounding regions are spheres, bounding boxes, or convex hulls. The tree structure is the key ingredient for the two-phase collision-checking process: In the first broad phase, the bounding regions for the root nodes of the trees are checked for collisions. If there is no collision, further collision checks on child nodes are unnecessary, as the root node covers the child nodes, and a collision can be ruled out. However, if the check on the root node detects a collision, the child nodes must be checked to determine if there is a collision. The checks are carried out recursively for the next tree level if one of the nodes in the upper level detects a collision.

Until now, we discussed if specific configurations are inside the free space C_{free} avoiding collisions, but not if the entire path connecting these configurations is also in C_{free}. The path connecting two intermediate configurations along a complete path is called *path segment*. A path segment is checked for collisions by sampling individual configurations on the segment and checking those for collisions. The sampling frequency Δq on the path segment is a hyper-parameter that should be chosen carefully. An obstacle could be missed if the parameter is too big; if it is too small, more time is used for unnecessary computation. One empirical method is to set Δq to a value smaller than the resolution of the configuration space, ensuring that the chosen configuration is not further away than two arbitrary samples in the configuration space. There are also algorithmic approaches for determining Δq: these rely on the robot's rigid body's geometric properties and upper bounds for their traveled distance for small movements in the configuration space. These bounds are dependent on the type of the manifold, as the movement characteristics

are different for translations (f.e. in $\mathbb{R}^2$) than for rotations (f.e. in $SO(2)$). Another critical decision is the order chosen for collision checks for the samples on the path segment. While the details are out of the scope of this work, it is essential to state that a sampling sequence with alternating order (e.g., *van der Corput sequence*) might detect collision on the path segment faster as it "jumps" back and forth between the start and end configurations at the beginning of its sampling phase.

We have outlined the foundations for understanding motion planning algorithms by explaining the essential concepts in the previous sections. In the following, we will shortly discuss two different sampling-based approaches for planning a path from start to end to complete the big picture. Path planning is sampling-based motion planning algorithms' third and last step, as stated in section 2.2.1.

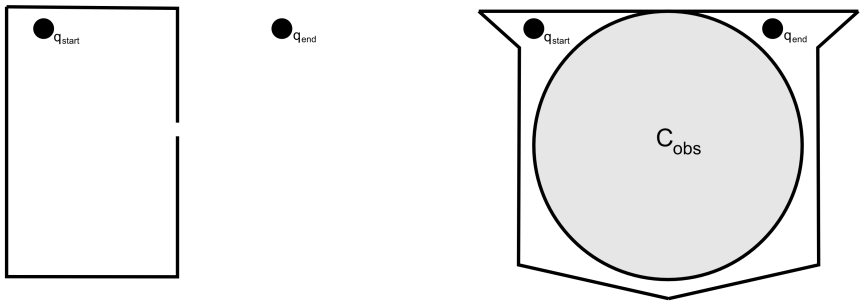
2.2.3 Single versus Multiple-query methods

Single-query and multiple-query planning algorithms are two different approaches for motion planning with random sampling. The single-query algorithm assumes the pair (q_{start}, q_{end}) of the robot to be given only once per robot and obstacle set, which means that no results from prior requests with start and pairs are assumed for the computations. This assumption means that the search graph is built from the ground up and exclusively for (q_{start}, q_{end}).

In contrast, the multiple-query model allows pre-computation and reuse of a search graph that can be used for new requests with pairs of robot start and end configurations.

2.2.3.1. Single-query motion planning

Single-query motion planning is executed by constructing an undirected search graph incrementally for a given configuration pair (q_{start}, q_{end}). The algorithm runs iteratively until it finds a collision-free path between q_{start} and q_{end}. The search graph can be started from either q_{start} or q_{end} or from both q_{start} and q_{end} simultaneously. Simultaneously constructing two search graphs is advantageous for situations where q_{start} or q_{end} are trapped within a cavity spot and hard to reach, a common issue for high-dimensional configuration spaces. Two cases where the construction of two graphs is beneficial are illustrated in Fig. 2.2. In Fig. 2.2 (a) the so-called *bug-trap* is illustrated. Here, a small opening needs to be passed to connect the two points. In Fig. 2.2 (b) the so-called *corridor* is illustrated. Here, the two points can only be connected by creating a search graph along the narrow corridor.

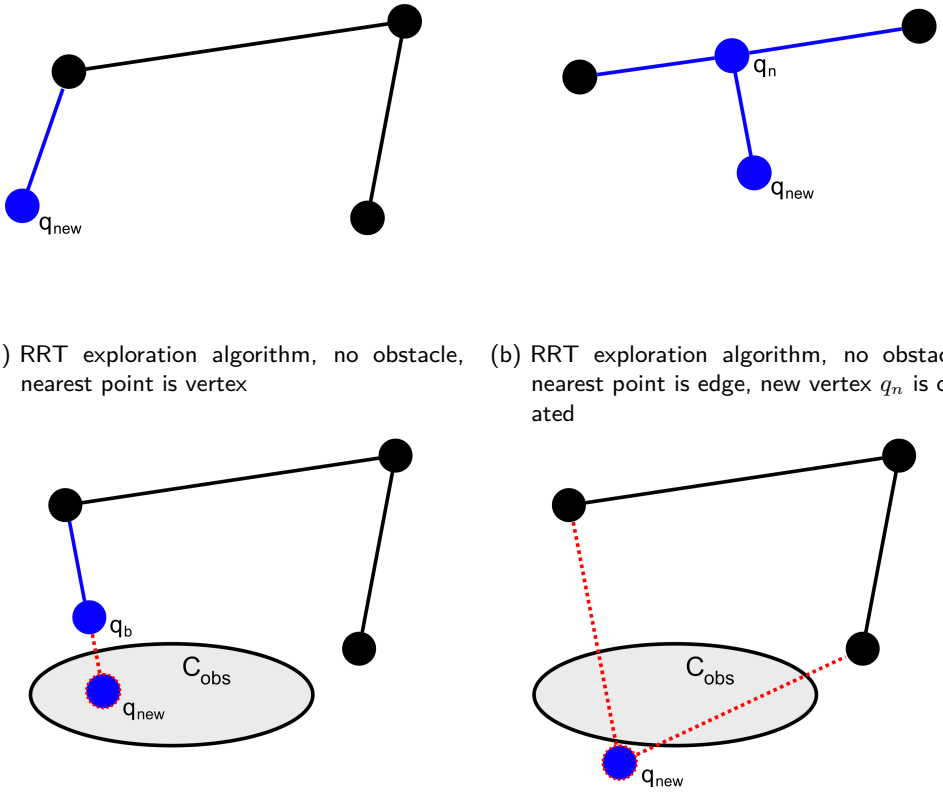


(a) bug-trap (b) corridor

Fig. 2.2.: Edge cases with exploration algorithms. In (a), a bug-trap is illustrated. Bug traps cause problems with unidirectional single-query methods where a single search tree is constructed from the start or end configuration. In (b), a corridor is illustrated. Corridors are hard to leave for multiple-query roadmap methods because the sampled configurations must all lie along the narrow corridor and be connected.

A critical step in the iterative process is the *local planning method (LPM)*: It tries constructing a path segment from the existing search graph to a newly sampled configuration q_{new} without causing a collision by employing the checks mentioned in section 2.2.2. The search graph is only extended if the sampled configuration is in the free space C_{free} and all points on the path segment from the search graph to the new point are collision-free, hence contained in C_{free}.

Several approaches exist for solving the graph extension problem (LPM). However, we will focus on a widely used method implemented in a variant of single-query motion planning methods called *rapidly exploring dense trees* (RRT). The graph extension in the LPM step is done by connecting the newly sampled point q_{new} to the nearest point q_n on the graph (see Fig. 2.3 (a)). The nearest point q_n might not be an existing vertex but could lie on an existing edge. In this case, q_n is inserted into the graph by splitting the edge and inserting q_n as a new vertex on the splitting location in an intermediate step before finally connecting q_{new} with it (see Fig. 2.3 (b)). Another issue that can arise during the LPM phase is that the newly sampled configuration may lie on an obstacle ($q_{new} \in C_{obs}$). In this case, the graph extension method mentioned above can be applied, but the new path segment from q_n to q_{new} should stop at the boundary of the collision object. A new vertex q_b can be inserted into the search graph as an endpoint for the new path segment at the stopping location instead of q_{new} (see Fig. 2.3 (c)). The newly sampled configuration q_{new} may also not be added to the search graph. The algorithm fails to add the new sample when q_{new} lies behind an obstacle that prevents a connection to the graph with a straight line (see Fig. 2.3 (d)).



(a) RRT exploration algorithm, no obstacle, nearest point is vertex

(b) RRT exploration algorithm, no obstacle, nearest point is edge, new vertex q_n is created

(c) RRT exploration algorithm, q_{new} hits obstacle

(d) RRT exploration algorithm, q_{new} hits obstacle

Fig. 2.3.: RRT exploration algorithm. The exploration algorithm extends the existing search graph with new collision-free samples in C_{free}. The search graph is used for single-query motion planning. The new configuration and the path segment connecting it to the nearest point on the search graph are checked for collisions with obstacles. Different cases for the exploration are illustrated.

The single-query planning approach attempts the actual path search for a given pair (q_{start}, q_{end}) by initializing the graph with q_{start} and incrementally expanding it with the previously mentioned exploration algorithm. Since q_{end} was not added to the graph in the initialization phase, the algorithm needs to periodically attempt a connection between q_{end} and the expanding search graph. The exploration algorithm could, for example, introduce a sampling bias towards q_{end} every n sampling iterations. After this attempt, the algorithm attempts a path search from q_{start} to q_{end}. If a path is found, the search succeeds, and no further exploration is necessary.

2.2.3.2. Multiple-query motion planning

The multiple-query motion planning algorithm expects the construction of an undirected search graph in advance before it can handle a planning request for a given configuration pair (q_{start}, q_{end}). The advantage of this approach is that the graph can be used for multiple requests once it is available. A trivial approach to building an initial search graph is constructing a grid with fixed resolution in the configuration space where each vertex is connected to its adjacent neighboring vertices. Vertices that would lie in the obstacle space C_{obs} are discarded. q_{start} and q_{end} are connected to the closest vertices on the grid. Depending on the size and location of the obstacles in C_{obs}, the graph might be sparse, but it can be extended during search requests. The grid resolution is a hyper-parameter that can be adjusted during the search: if a search attempt fails, the algorithm can double the grid resolution and reattempt the path search. These graphs are also called *roadmaps* in the robotics literature. In the following, we will discuss a more generic and flexible roadmap method that can be used in a broader range of applications, the *sampling-based roadmaps*.

Sampling-based roadmaps are constructed with sampling methods similar to the exploration algorithm described in section 2.2.3.1. In contrast to the previously mentioned grid structure, the sampling-based roadmap relies on a dense sequence that creates new configurations q_{new}. The algorithm checks q_{new} for collisions and adds it to the search graph. In the next step, a neighbor selecting scheme is applied to create path segments for increasing the number of neighbors of q_{new}. Various schemes are available, e.g., radius-based, nearest-k points, and visibility. In Fig. 2.4, the radius-based scheme tries to connect all vertices within a ball of the given radius to q_{new}.

A very efficient version of sampling-based roadmaps is the visibility roadmap. This variation tries to connect a newly sampled configuration q_{new} to all vertices in the search graph without prior filtering. However, it only keeps q_{new} if it is connected to two unconnected graph components. Because only then is it assumed that the new configuration increases the *visibility* of the existing search graph. In Fig. 2.5, the visibility concept is displayed for a single node in the presence of a single obstacle.

2.3 Forward Kinematics

In this section, we will introduce the *forward kinematics* (FK) function for mapping a given configuration of the robot to the task-space. In contrast to the inverse

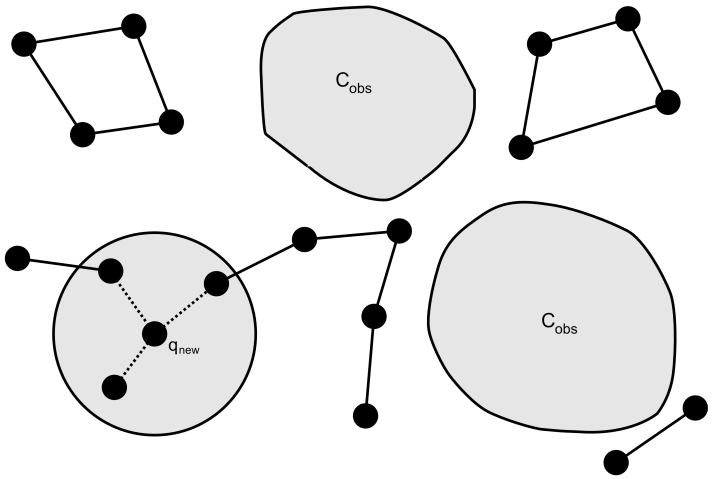


Fig. 2.4.: Sampling-based roadmap. Construction. The roadmap is constructed by sampling configurations q_{new} from a dense sequence. After performing collision checks for each segment, the new configuration is connected to its neighbors with new path segments. The neighbor selection scheme in this illustration is based on a fixed distance threshold from q_{new}.

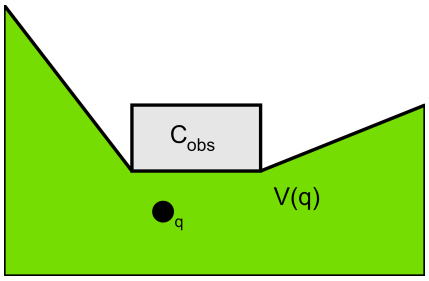


Fig. 2.5.: Sampling-based roadmap. Visibility. The green area visualizes the *visible* part of the configuration space from the sampled configuration q's perspective. Visible in this context means that any point in the green area can be connected with a path segment to q without hitting the obstacle C_{obs}.

kinematics function, which will be introduced in section 2.4, the FK is well-defined for all types of joints mentioned above and delivers a unique, exact, and correct output for the provided input configuration.

In the big-picture of robot motion planning, several use cases exist for the FK function. For example, iterative methods for inverse kinematics utilize the FK for iteratively calculating task-space positions for intermediate solutions and comparing them to the input of the IK, which is the desired position of the end-effector. This work will focus on defining the FK for *open-loop kinematic chains*. In the following, we will briefly introduce the kinematics problem based on the detailed study of this problem in [Cra05, p.62-75].

The study of kinematics encompasses all geometric and time-based motion properties without notice to the forces that cause the motion. In this brief introduction to kinematics, we restrict ourselves to determining the position and orientation of robotic manipulator linkages in static situations. To achieve this goal, we fix frames to mechanical parts and describe the relationship between frames and how they change with the robot's motion. More specifically, we narrow this problem down to ultimately finding the position and orientation of the robotic arm's end-effector relative to the base as a function of its joint angles.

Linkages (links) in a robotic setting describe a rigid body that defines a relationship between two neighboring joint axes of a manipulator. We will introduce the Denavit-Hartenberg (DH) method for describing the kinematic relationship between linkages. The DH convention can describe this kinematic relationship with only four parameters, reducing the six degrees of freedom (three axes of rotation, three axes of translation) without introducing restrictions.

The four parameters that describe the kinematic relationship for a link between two joints with the DH convention are:

- a: link length,
- α: link twist,
- d: link offset, and
- θ: joint angle.

Fig. 2.6 illustrates these parameters on two revolute joint axes i and $i+1$ that are connected by the link $i-1$.

The difference in these four parameters for revolute and prismatic joints is that for revolute joints, d is fixed, and θ is variable, whereas for a prismatic joint, θ is fixed, and d is variable. We will attach a frame to each link of the robotic manipulator:

- $\mathbf{Z_i}$ is coincident with joint axis i,
- $\mathbf{X_i}$ is the perpendicular axis lying on the shortest path between the joint axes Z_i and Z_{i-1},
- $\mathbf{Y_i}$ is inferred with the right-hand rule for coordinate axes, and
- the origin of frame i is located at the intersection of X_i with Z_i.

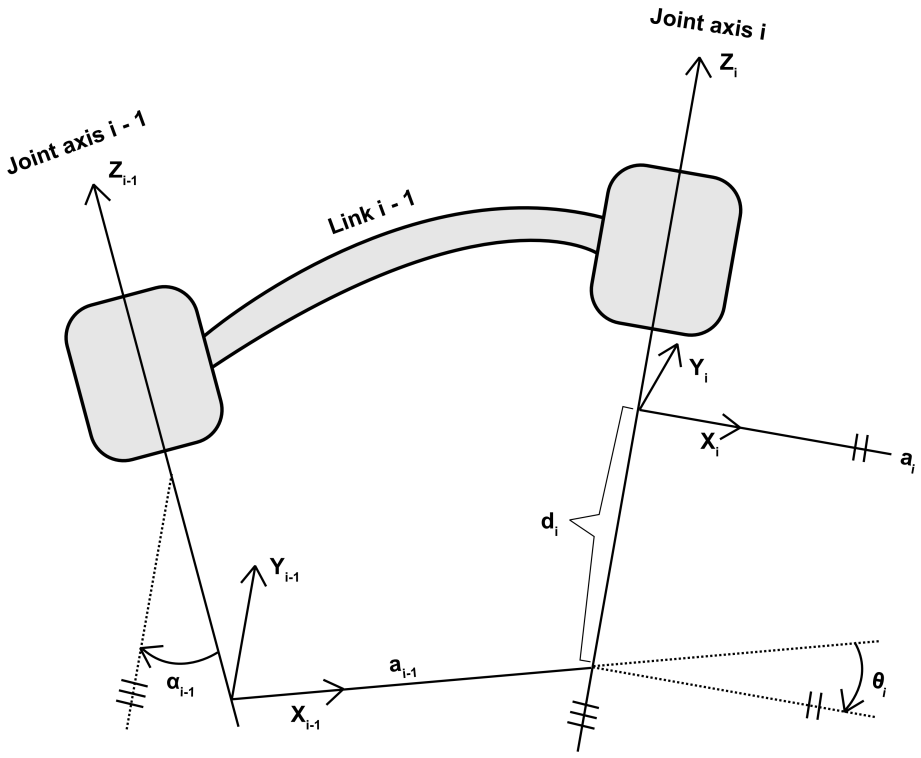


Fig. 2.6.: Link Frames in Denavit-Hartenberg convention. We illustrate the Denavit-Hartenberg (DH) convention for attaching frames to a robotic manipulator's links. The DH convention is a widely used method for calculating the forward kinematics of a robotic arm.

We can now derive the link transformations from frame i to $i-1$ and from n to 0 (end-effector to base). The link transformations are a function of one variable since the other three variables of the DH convention are fixed by the mechanical design of the link. The transformation ${}^{i-1}_{i}T$ from link frame i to $i-1$ is a subproblem in the big picture, where we want to calculate the end-effector's position and orientation relative to the base. The subproblem ${}^{i-1}_{i}T$ can be broken down into four separate subproblems given the DH parameters of the robotic manipulator, one for each DH parameter:

- **R**: rotation about X_{i-1} by α_{i-1},
- **Q**: translation along X_R by a_{i-1},
- **P**: rotation about Z_Q by θ_i, and
- **i**: translation along Z_P by d_i.

The resulting transformation matrix from link i to $i-1$ is denoted in equations 2.3 to 2.5, where D_a stands for a translation along axis a and R_a for a rotation about a.

$$ {}^{i-1}_{i}T = {}^{i-1}_{R}T\,{}^{R}_{Q}T\,{}^{Q}_{P}T\,{}^{P}_{i}T \tag{2.3} $$

$$ {}^{i-1}_{i}T = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)D_z(d_i) \tag{2.4} $$

$$ {}^{i-1}_{i}T = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5} $$

The transformation from the end-effector (n) to the base (0) of the robotic arm can be calculated by concatenating the transformation matrices between neighboring links beginning with the last link:

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \ldots {}^{N-1}_{N}T \tag{2.6} $$

Equation 2.6 will be a function of all n variables, one joint angle per revolute joint.

2.4 Inverse Kinematics

In the following, we will briefly introduce the inverse kinematics problem based on the detailed study of this problem in [Cra05, p.102-128].

The *inverse kinematics* (IK) of a robotic manipulator calculates the set of joint angles that achieve the desired position and orientation of the end-effector. The IK has a non-linear solution since we are trying to solve for the joint angles in equation 2.5, which are inside the non-linear trigonometric sine and cosine functions. For example, the link transformation equation (see eq. 2.5) of a six DoF robotic arm has 12 variable numeric values (the last row is fixed), which would result in 12 equations and six unknowns (6 joints angles). The rotational entries in eq. 2.5, however, are not independent; they correspond to only three independent equations

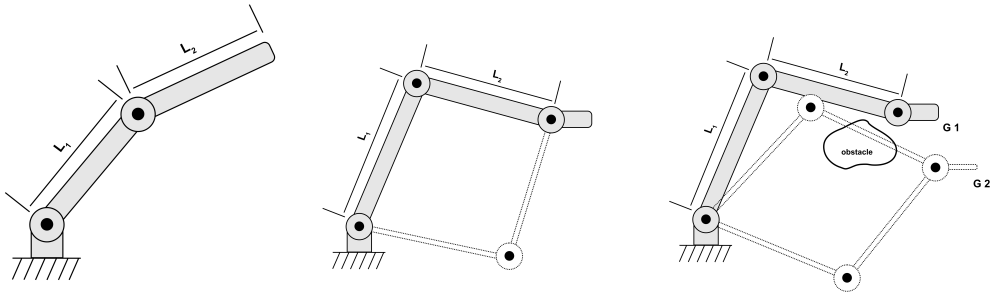
(rotation about three axes). The dependence of the rotational entries leaves us with six equations and six unknowns, where the equations are non-linear.

We need to check the non-linear system of equations for

- the existence of solutions,
- the uniqueness of solutions, and
- an appropriate solution method.

2.4.1 Existence of solutions

The solution for the inverse kinematics problem of a robotic manipulator lies in its workspace. The workspace of a robotic arm can be categorized into two types: the dextrous and the reachable workspace. The **dextrous workspace** contains all points in the reachable space of the end-effector that the arm can reach with all possible rotations. The **reachable workspace** contains all points in the reachable space of the end-effector that the arm can reach with at least one rotation. The definitions show that the dextrous workspace has stricter requirements than the reachable workspace.



(a) 2R robot (b) 3R robot - multiple solutions (c) 3R robot - multiple solutions and obstacle

Fig. 2.7.: Inverse kinematics robot schematics. We illustrate robotic manipulators and possible scenarios for their inverse kinematics. A robotic manipulator with two rotational joints is illustrated in (a). In (b), two solutions are illustrated for a robotic manipulator with three rotational joints. In (c), two solutions **G1** and **G2** are illustrated for a robotic manipulator with three rotational joints. Solution **G1** is invalid as it collides with an obstacle encoded in the configuration space.

For the 2R robotic arm in Fig. 2.7 (a), when $l_1 == l_2$, the dextrous workspace only contains the frame origin of the base joint as only that point can be reached by the end-effector with all possible rotations. In contrast, the reachable workspace is a disk of radius $2l$ with its center at the base joint. However, when $l_1 \neq l_2$, the dextrous

workspace is empty, and the reachable workspace is a ring with outer radius $(l_1 + l_2)$ and inner radius $|l_1 - l_2|$.

When a manipulator has less than six degrees of freedom (DoF), it cannot reach general goal positions and orientations in 3D space, as the system of equations for the IK becomes underdetermined. For manipulators with less than six DoF, the IK should try to determine the nearest reachable goal instead of the desired goal.

2.4.2 Multiple solutions

When solving the inverse kinematics of a robotic manipulator, multiple solutions are possible in specific scenarios. The critical question is, which solution should the IK choose? For example, the 3R robotic arm in Fig. 2.7 (b) has multiple solutions for the same goal in its operating plane. The system has to choose from multiple solutions. One strategy is to choose the closest solution to the current configuration of the arm in the configuration space. Considering the link lengths for this approach with a suitable weighting scheme is essential, as moving smaller links costs less execution time. Farther solutions might still be selected when the closest solution interferes with obstacles in the configuration space. We illustrated this situation in Fig. 2.7 (c) where moving from goal *G 1* to *G 2* with the closest solution interferes with an obstacle. For this reason, calculating all solutions must be possible with the IK in the presence of obstacles.

In general, the number of solutions depends on the DoF count of the manipulator, the number of fixed DH parameters, and the allowed (rotation) limit for the joints. More non-zero DH link parameters lead to more solutions as, for example setting the link length $a \neq 0$ moves the neighboring joint axis out of the plane of the previous joint axis. The combination of a non-zero offset a and non-zero link twist α allows the manipulator to reach a greater volume by moving "out of the plane" and therefore attain more solutions for a particular goal.

2.4.3 Method of solution

There is no general solution algorithm for the non-linear IK equations. The solution to the IK problem is to find **all** joint angles that reach a given goal with the given position and orientation. The solution can be calculated with numerical iterative

solution methods or closed-form expressions. Iterative solvers are slower than closed-form solution approaches. In contrast, closed-form expressions are only possible for manipulators with specific mechanical properties.

Closed-form approaches can be subdivided into **algebraic** and **geometric** methods. Algebraic solutions rearrange the given equations to a form where a solution for the joint angles exists. In contrast, geometric methods decompose the arm's spatial geometry into several plane-geometry subproblems. These subproblems can be solved for the joint angles with tools of plane geometry. For example, a 3R robotic arm that operates in a single plane has its angle twist DH parameter α set to 0. The planar operation allows us to solve for the joint angles with analytical expressions more easily.

3 Fundamentals of X-ray Computed Tomography

In this chapter, we will first introduce the fundamental concepts of X-rays and explain the effects that lead to their formation. Subsequently, we will introduce the fundamental acquisition geometries for X-ray imaging. Finally, we will provide a problem statement for the computed tomography problem.

3.1 Fundamental concepts of X-rays

X-rays are electromagnetic radiation ranging from 3×10^{16} to 3×10^{19} Hz. The frequency specifies the number of vibrations of the electromagnetic wave per second. For reference: the frequency of light that is visible to the human eye ranges from 420×10^{12} to 750×10^{12} Hz, and radio waves that are transmitted for FM and AM radio stations range from 300×10^{6} to 300×10^{9} Hz.

X-rays can be produced with an X-ray source that emits electrons by heating up a thin wire coil that acts as a cathode. The released electrons hit an anode that absorbs their kinetic energy. In electron absorption, two effects produce X-rays: *Bremsstrahlung* and *characteristic radiation*. [HJL21]

Bremsstrahlung X-rays are emitted as Bremsstrahlung when the negatively charged electron passes through the nucleus of an atom at the anode and gets deflected and decelerated by the positively charged nucleus. The emitted X-ray's energy corresponds to the incoming electron's kinetic energy loss.

Characteristic radiation X-rays are emitted in the nucleus of the atoms at the anode when the incoming electron hits an electron located in the atom's inner shell. The collision ejects the hit electron from the atom's shell, and the free slot is occupied by an electron from the outer shell of the atom nucleus. X-ray radiation is produced by the electron moving from an outer shell to the free spot at the atom's inner shell. The number of traversed shells determines the energy of the emitted X-ray.

In the following, we will briefly outline three essential concepts for the interaction of X-rays with matter. These concepts are essential, as they affect the measurement process: X-ray photons that do not hit the detector do not appear on the resulting detector image.

Photoelectric effect The photoelectric effect is caused by X-rays hitting matter (e.g., the measured sample) and ejecting electrons bound in the matter's atomic nuclei. The probability of this interaction increases when the binding energy of the electrons in the atomic shell is slightly above the interacting X-ray photon's energy. This relationship also means that materials with a low atomic number are practically transparent to X-rays as the difference between their electron's binding energy and the energy range of typical X-rays is too high. The high gap between these energies omits the photoelectric effect.

Rayleigh scattering For atoms with high atomic numbers, the incoming X-ray photon cannot eject the electron sitting in the atom's shell due to the high binding energy of the atomic nucleus. The X-ray photon is scattered after hitting the electron without losing kinetic energy and continues on a different path.

Compton scattering Compton scattering describes the combination of the photoelectric effect and Rayleigh scattering: The incoming X-ray can eject an electron from an atom's shell after hitting it. Releasing the electron causes the emission of a new X-ray through characteristic radiation. In contrast to the photoelectric effect, the X-ray photon continues along a different path since its kinetic energy was not fully absorbed by the electron it hit.

3.2 Fundamentals of X-ray Acquisition Geometries

This section will introduce the most important X-ray acquisition geometries and the underlying concepts based on [HJL21]. The actual signal measurement process at the detector is outside this thesis's scope.

Lambert-Beer Law The Lambert-Beer law models the interaction of X-rays with the matter by modeling the attenuation of rays passing through matter. The law states that the change in the ray's intensity is proportional to the thickness and the concentration of the material it passes through. More specifically, the X-ray's intensity will decay exponentially, which will depend on its incident intensity and the material's attenuation coefficient. For inhomogeneous matter, the attenuation of a ray is position dependent since the attenuation coefficient depends on the material composition and will be different at different positions.

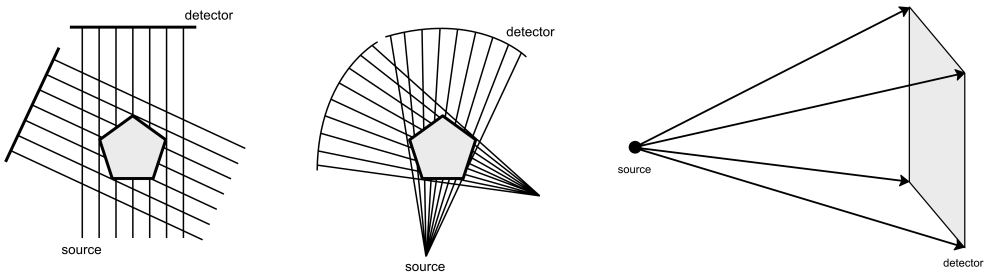
One fundamental assumption of the Lambert-Beer law is that the X-ray source and the detector elements that convert the measurement to digital signals are infinitely thin in their dimensions. This assumption does not hold in practice; hence, we must introduce discrete measurement geometries. In the following, we will introduce the most common types of acquisition geometries.

Radon transform The Radon transform is a fundamental concept for reconstructing a sample from measurements with X-rays. The Radon transform states that perfectly reconstructing the scanned sample is possible when it is measured from a complete set of line integrals over all angles. At each angle θ, a set of parallel line integrals passes through the sample and hits the detector. The resulting set of line integrals is called the projection of the sample at angle θ.

Parallel-beam geometry The parallel-beam geometry setup consists of an X-ray source and a detector that can be translated and rotated. In the measurement process, a single line integral can be measured at the detector before the source-detector pair needs to be translated to measure a line integral parallel to the previous one. This process is repeated after rotating the source-detector pair for additional line integrals. The parallel-beam geometry is illustrated in Fig. 3.1 (a), where multiple line integrals are depicted at once. The measurement process is very time-consuming, so new methods were developed for more efficient acquisition of samples.

Fan-beam geometry The fan-beam geometry consists of an X-ray source and a detector, but in contrast to the parallel-beam setup, the detector is arranged with a curved array of elements, and the source emits a divergent beam of X-rays (see Fig. 3.1 (b)). This arrangement allows data acquisition from multiple line integrals in a single position of the source-detector pair. Hence, the introduction of the fan-beam allowed for improvements in measurement time.

Cone-beam geometry The cone-beam geometry improves measurement times even further compared to fan-beam setups by replacing the single-dimensional detector with a curved or planar two-dimensional detector (see Fig. 3.1 (c)). The 2D detector allows the acquisition of multiple rows of line integrals at once, eliminating the need for translating the source-detector pair in the object's z-direction.



(a) Parallel-beam (b) Fan-beam (c) Cone-beam

Fig. 3.1.: X-ray acquisition geometries. We illustrate three acquisition geometries for X-ray computed tomography. The parallel-beam (a) and fan-beam (b) setups acquire measurements row by row. With the cone-beam setup (c), the entire sample volume can be measured at once at the detector.

3.3 Fundamentals of CT Reconstruction

In the computed tomography (CT) problem, we try to recover a function $f : V \to \mathbb{R}$. The function f assigns a real-valued absorption coefficient to every point in a volume $V \subset \mathbb{R}^n$. For X-ray CT specifically, this real value is the X-ray attenuation coefficient of the measured sample at the given point. This section will outline a solution approach for solving this problem, the *series expansion* approach.

In the series expansion approach, we solve for the function f with the following three steps.

Discretization First, we discretize f by choosing a finite set of *basis functions* $b_i : V \to \mathbb{R}$, $i \in I$. Strictly speaking, b_i are not basis functions of f since their linear combination is not guaranteed to represent every possible f exactly. To approximate f on a computer with reasonable visual quality, we can assume that b_i does not need to be real basis functions. We define

$$\hat{f}(\cdot) = \sum_{i \in I} x_i b_i(\cdot), \qquad \{x_i\}_{i \in I} \subset \mathbb{R}, \tag{3.1}$$

such that $||f - \hat{f}|| < \varepsilon$, where $\varepsilon > 0$ small. The x_i represent the attenuation coefficients of the discretized volume. If the two-dimensional volume is approximated on a $k \times k$ grid with $I = \{1, ..., k^2\}$, we can choose the following basis functions

$$b_i(c_1, c_2) := \begin{cases} 1 & \text{if } (c_1, c_2) \text{ is inside } i\text{–th pixel} \\ 0 & \text{else.} \end{cases} \tag{3.2}$$

Measurement model The measurement model describes how the signals $m = (m_j)_{j \in J} \subset \mathbb{R}$ on the detector could be generated by the measurement process that sends X-rays through the volume f with

$$\mathcal{M}_j : (f : V \to \mathbb{R}) \longrightarrow \mathbb{R}, \tag{3.3}$$

where $\mathcal{M}_j f = m_j$ for all $j \in J$. If we assume that $\mathcal{M}_j$ is linear, we can apply it to the individual "basis functions" in the sum inside $\hat{f}$:

$$\mathcal{M}_j f \approx \mathcal{M}_j \hat{f} = \mathcal{M}_j \left(\sum_{i \in I} x_i b_i \right) = \sum_{i \in I} x_i \, \mathcal{M}_j b_i \tag{3.4}$$

For X-ray CT, the model $\mathcal{M}_j$ could be a line integral along the ray beginning from the X-ray source traversing the volume f and hitting the detector pixel j.

We can form the system of equations

$$Ax = m, \tag{3.5}$$

where the entries a_{ji} in the system matrix A represent the application of the measurement model on the individual "basis functions" b_i inside the approximated volume $\hat{f}$: $a_{ji} = \mathcal{M}_j b_i$. In other words, the rows of matrix A represent individual rays passing the volume defined by x, and each entry m_j on the right-hand side represents the measurement of a particular pixel on the detector.

Solution We can now compute an approximate solution for $\hat{x}$ such that

$$\hat{f}^* = \sum_{i \in I} \hat{x}_i b_i \tag{3.6}$$

and $\hat{f}^*$ is the desired reconstruction of the measured volume f that contains the sample.

For the image reconstructions of our experiments in chapters 5, 6, and 7, we always used iterative methods. Other alternatives as analytical solutions or filtered back-projection (FBP), were not viable for our scenario. We never considered analytical solutions as the system of equations is under-determined, and the problem is ill-posed in general, meaning that a slight variance in the detector measurements might lead to a significant change in the solution for analytical approaches. FBP was not a viable option as it requires measurements from the entire angular range, which is against the nature of installing the robotic arm in a restrictive environment (see unsampled spots in chapter 6, Fig. 6.2) and also against our problem statement of sparsely sampling "valuable" spots on the spherical trajectory (see chapter 7).

We used the iterative conjugate gradient solver for the image reconstructions in chapters 5, 6, and 7. We fixed the number of iterations to 30, as more iterations did not improve the cost function. The solver ran on a Tikhonov regularized weighted least squares problem [Wil79] with the Josephs method for discretizing the X-ray transform [Jos82] and cone-beam geometry. We used our C++ reconstruction framework *elsa* [LHF19] to perform the reconstructions. The reconstruction volume consisted of $720 \times 720 \times 720$ isotropic voxels with a $38\ \mu m$ spacing for all our experiments.

Part II

X-ray Computed Tomography with a Robotic Sample Holder

4 Introduction to Lab Setup with Robotic Sample Holder

4.1 Hardware Setup

4.1.1 X-ray Laboratory Overview

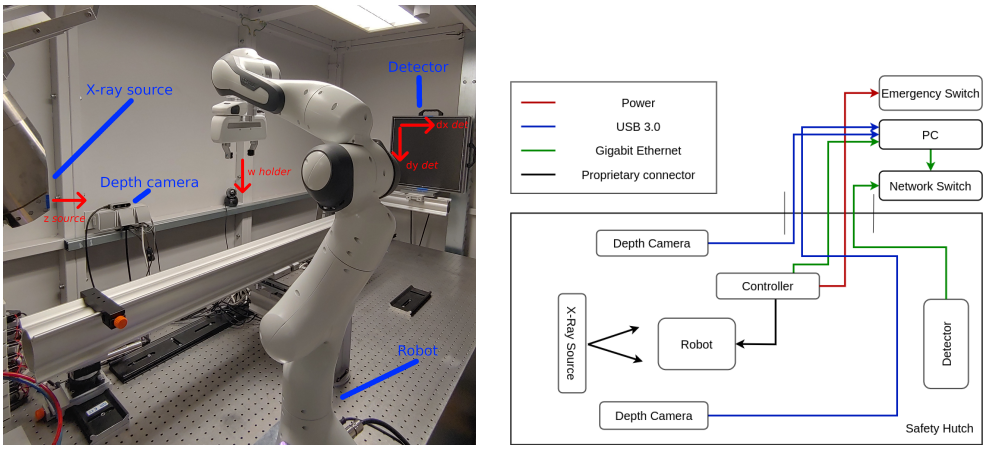
The X-ray laboratory is an experimental setup for research purposes divided into two areas: a safety enclosure and the operator's desks outside. The following paragraphs are an adapted excerpt from our publication [Pek+22b] about the hardware setup in our X-ray CT laboratory. We added information about the changes we made for our subsequent publication [Pek+23a].

The system's hardware components are displayed in Fig. 4.1. The main difference to a conventional X-ray CT setup is the seven degrees of freedom robotic arm *Panda* from the manufacturer FRANKA EMIKA [@EMI20]. It has a maximum reach of 855 mm and a repeatability of 0.1 mm when repeatedly moved from a specific starting pose to a goal pose. It has two fingers that can move on a fixed axis and grasp objects. The maximum allowed payload is 3 kg. The robotic arm and the depth cameras are connected directly to a computer, while the detector is accessible through a network interface. The robotic arm can be turned off in case of emergency from outside of the safety enclosure with a power switch (see fig. 4.1b).

Two *Intel Realsense D435* depth cameras capture the robot's movements and provide 3D information about the surroundings as a point cloud. The cameras are connected directly to the workstation and are used for the collision detection mechanism described in section 4.2.3.2.

The robotic arm is mounted on a table inside a safety enclosure for X-ray CT, which houses the X-ray source and the detector (see fig. 4.1a). The detector has a maximum resolution of 2880x2880 and is connected to a different workstation on the network, which provides a network interface for triggering image capturing. Our workstation retrieves the raw 16-bit grayscale image over the network.

We moved the robotic arm further away from the X-ray source and closer to the detector for the experiments in our second submitted manuscript [Pek+23b] (see Fig. 6.1a). Changing the arm's position increased the field of view for the experiments and allowed the use of a sample holder with greater dimensions and different gripper shapes for added flexibility.



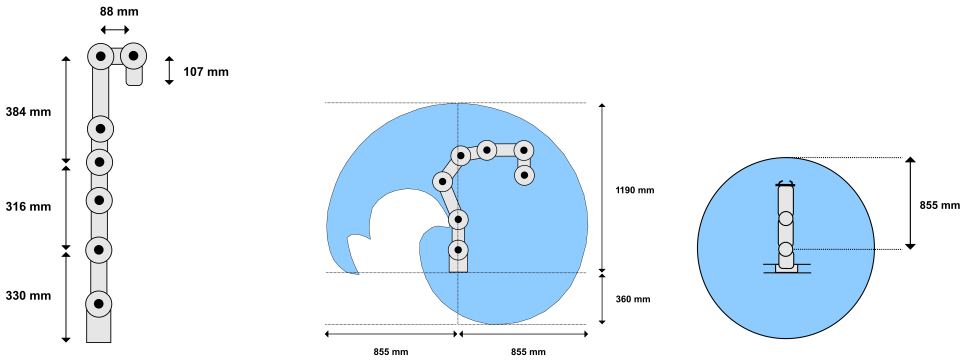
(a) Lab photo (b) Hardware schematics

Fig. 4.1.: Hardware setup. In Fig. 4.1a, the robotic arm is mounted on a table with the source and the detector inside a safety hutch. The source-to-robot distance is 40 cm, and the robot-to-detector distance is 176 cm. Two depth cameras monitor the movement of the robot and send a stop signal to the robot controller when the executed trajectory interferes with obstacles. The robotic arm can also be stopped by a manual power switch routed to the operator table outside the hutch (see fig. 4.1b). The relevant coordinate systems are visualized in red in Fig. 4.1a. The x and y-axis are determined by the right-hand rule.[**Pekel_2021**]

4.1.2 Robotic Arm

Throughout this Ph.D. project, we exclusively used the FRANKA EMIKA Panda robotic arm in both the laboratory and the simulation environment (see figures 4.1 (a) and 4.6). Panda is offered at an attractive price (below 30.000 EUR), provides easy integration with the *ROS* development environment, and the manufacturer provides the 3D designs of the individual links of the robotic arm as CAD files. We used the CAD files for the active collision detection algorithm (see sec. 4.2.3.2). However, they can also be included in the simulated measurements with the tomographic imaging library for more realistic measurements.

The kinematic properties and dimensions of the Panda arm are illustrated in the figures 4.2 (a) - (c). We extracted the data for the figures from the official specification document for Panda [@EMI20] and the *franka_ros* documentation [FRA23].



(a) Panda - model (b) Panda - reach (side view) (c) Panda - reach (top view)

Fig. 4.2.: FRANKA EMIKA Panda robotic arm. (a) depicts the dimensions of the FRANKA EMIKA Panda robot. (b) and (c) depict the arm's reach from the side and top view.

The arm has seven degrees of freedom (DoF) with seven rotational joints. In Fig. 4.2 (a), we can see the dimensions and composition of Panda's links. Its compact dimensions (0.384 m max. link length) make it ideal for use in a space-restricted environment, like the safety enclosure in our X-ray CT laboratory. In Fig. 4.2 (c), we can see that the arm can reach all points within its maximum reach of 855 mm to its sides with at least one rotation. Fig. 4.2 (b) displays Panda's sideway reach for a static configuration. We can see that it can not cover the area in its rear, which can be better reached by rotating the arm at its first joint attached to the base.

4.1.3 Sample Holder

In this section, we will discuss the sample holder part and the ideas that lead to the final design in our publications [Pek+22b; Pek+23b].

The necessity for introducing a sample holder part originated from the repeatable placement error of the FRANKA EMIKA Panda robotic arm in our laboratory setup. We intended to calibrate the exact position and orientation of the sample on the detector images with a suitable calibration structure. Another reason for introducing an intermediate item between the arm's end-effector and the sample is the added flexibility for sample mounting. Without the intermediate sample holder item, the robotic arm needs to pick the sample up with its two fingers.

We decided on the calibration structure in the initial design and prototyping phase of the sample holder item. After initial attempts with non-parametric arrangements

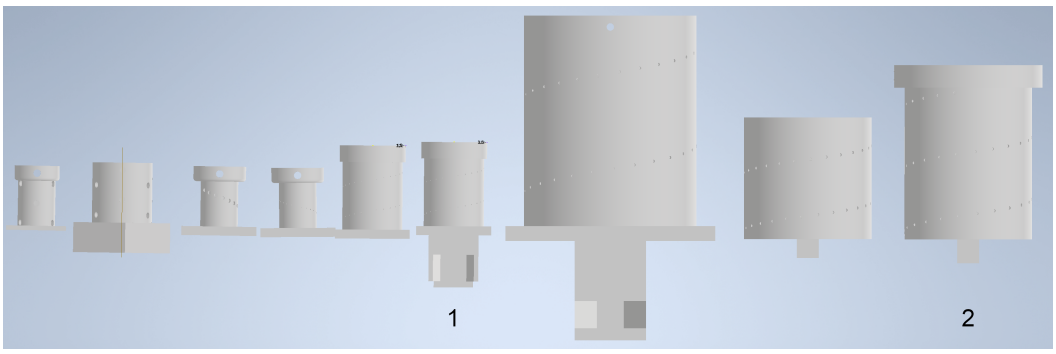


Fig. 4.3.: Sample holder parts. We illustrate the most important sample holder parts we designed throughout this Ph.D. project from the first (left) to the last (right). We used part no. 1 for the experiments in our first publication [Pek+22b] and part no. 2 for the experiments in our second publication [Pek+23b]. The cylindrical body houses a helical calibration structure that is used to calibrate the orientation and position of the sample on the individual acquisition images.

of spheres on a cylinder, we have decided to use a helix structure with a well-defined parametric equation. The helix structure can be calibrated on the 2D detector images given the X-ray acquisition geometry and the helix's equation. We placed aluminum spheres with high absorption coefficients into the holes and used a suitable circle segmentation algorithm. The parametric equation can be used throughout the calibration pipeline, beginning with the design process where the helix is modeled on a cylinder in our design software. The modeling process is followed by the manufacturing procedure, where a suitable machining process needs to place the holes in the cylinder without violating the parametric equation of the helix. In early iterations of our prototyping process for the sample holder item, we used a 3D printer. However, the holes in the resulting part were placed inaccurately. The holes on the helix deviated by up to 0.1 mm from their intended position, affecting calibration results [Pek+22b]. For our final design, we cut the holder from a solid piece of PVC and drilled the calibration holes with a precision cutter (CNC) machine. The machine received the exact locations of the holes as 3D coordinates and cut them individually into the cylinder that resulted from the first cutting step. We illustrated the unique calibration structure designs introduced throughout this Ph.D. project in Fig. 4.3.

In subsequent iterations of our sample holder item designs, we split the structure into the calibration and gripper parts for more time and cost-effective prototyping. With the introduction of different gripper shapes, we maximized the potential reach of the robotic arm: in [Pek+23b], we demonstrated that up to 99.1% of all possible rotations can be reached by the robotic arm when using the l-shaped gripper and removing the restrictions introduced by installing the arm in the restrictive safety

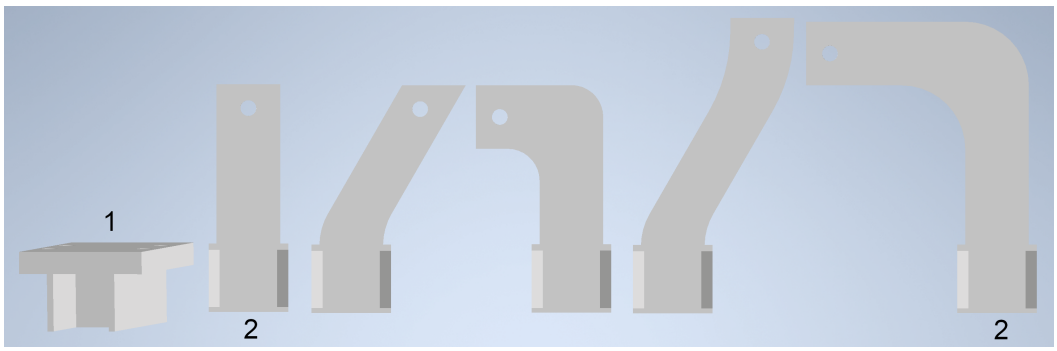


Fig. 4.4.: Sample holder part grippers. We illustrate the unique gripper designs for the sample holder part that we designed throughout this Ph.D. project from the first (left) to the last (right). We used gripper no. 1 for the experiments in our first publication [Pek+22b] and the two grippers labeled no. 2 for the experiments in our second publication [Pek+23b]. The l-shaped gripper part allows maximal coverage of all sample orientation (99.1%, no obstacles encoded) when attached to the sample holder part no. 2 from Fig. 4.3.

enclosure. The most unique gripper shape designs introduced throughout this Ph.D. project are illustrated in Fig. 4.4.

4.1.4 Depth Cameras

We have used two Intel Realsense depth cameras (see Fig. 4.5) for the active collision detection algorithm outlined in section 4.2.3.2. The Realsense camera combines data from two RGB sensors and an infrared projector to capture depth information about a scene. An additional RGB sensor is provided for colored pictures and videos of the captured scene. The maximum resolution of the depth stream is 1280×720 at 90 frames per second (fps). The colored video stream has a 1920×1080 resolution at 30 fps [@Int23a]. The advantage of the Intel Realsense platform is the smooth integration within the ROS ecosystem. We used the *realsense-ros* ROS packages [@Int23b] that Intel provides to read the point clouds through an abstract interface and with ROS-native data types.

Fig. 4.5.: **Intel Realsense depth camera.** We used two Realsense cameras for the active collision detection algorithm described in section 4.2.3.2. The maximum resolution of the depth stream is 1280×720 at 90 fps and the colored video stream has a 1920×1080 resolution at 30 fps.[@Int23a]

4.2 Software Components

4.2.1 Simulation Environment

The simulation environment served two primary purposes during this Ph.D. project. First, it was a testing environment for new algorithms and the robotic arm's control routines. We validated the critical safety components of our software package with the simulation environment, where we could simulate the FRANKA EMIKA Panda robotic arm. Second, we saved measurement time for our experiments, as it was possible to simulate measurements with the simulation environment as we mirrored the robotic arm's environment in the laboratory within the simulation. We could also plan trajectories in the simulation environment before they were executed in the laboratory, saving us up to 45 minutes of planning time.

The simulation environment is based on *Gazebo* [KH04] and is depicted in Fig. 4.6. The robotic arm is placed on a table captured by two depth cameras from the top and one RGB camera from the side. The depth cameras are used for testing the active collision detection algorithm described in section 4.2.3.2. The RGB camera was used in the early stages of the project for prototyping different calibration algorithms based on camera images but was later replaced by the tomographic imaging library *elsa* [LHF19]. For visualization, a sample holder is attached to the robotic arm's end-effector (blue cuboid) in Fig. 4.6.

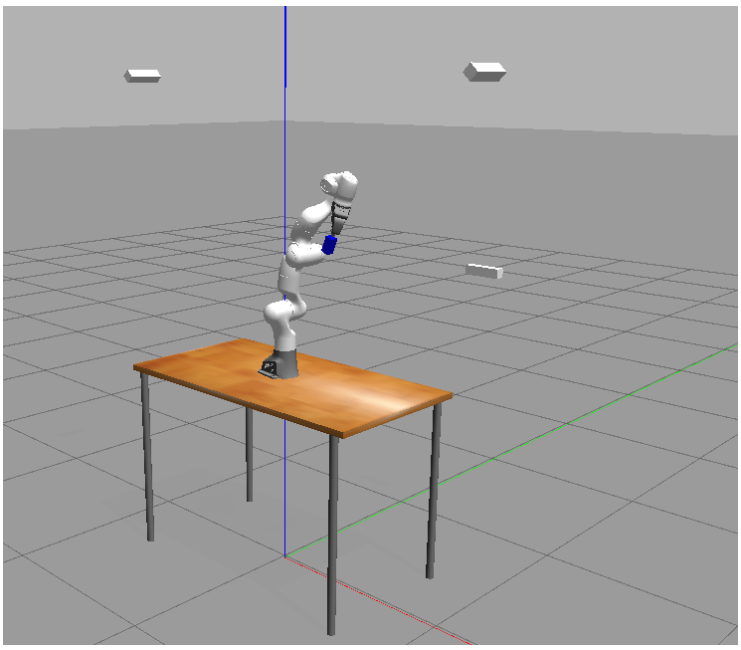


Fig. 4.6.: Simulation environment. Our Gazebo [KH04] simulation environment includes the robotic arm, two depth cameras, and one RGB camera. The depth cameras are used for testing the active collision detection algorithm. The RGB camera was used for testing new calibration algorithms in the early phases of this project. We use *Gazebo* for simulating the dynamics of the robotic arm.

4.2.2 System Architecture

In this section, we will outline the system architecture of our software project for the robotic sample holder [Pek21]. The project is organized into the packages core, utilities, computed tomography, safety, messages, and web.

In the following sections, we will outline the functionality provided by each package in detail. In Fig. 4.7, we provide an overview of all packages and their components.

4.2.2.1. Core

We implement the most important functionality in the core folder. In the following, we provide an exhaustive list of the core components and their functionality:

Experiment The experiment component is the entry point to starting new experiments or resuming interrupted experiments. This process includes sample acquisition, image segmentation, calibration, and computing a reconstruction volume.

These individual steps of the experiment pipeline can be run independently when the experiment identifier (ID) is provided.

OnlineExperiment The online experiment component differs from the experiment component in its operating mode: It generates and executes the acquisition trajectory one way-point at a time. Hence, it cannot perform the segmentation and calibration steps independently, except for the final reconstruction volume. We implemented this component for the trajectory optimization algorithm described in chapter 7.

TrajectoryGenerator The trajectory generator can generate two types of trajectories (circular and spherical) starting from the current configuration of the robotic arm. The trajectory is generated based on the parameters provided by the user on the user interface.

TrajectoryManager The trajectory manager is responsible for the efficient reuse of calculated trajectories. It calculates a unique identifier for every trajectory that is requested by the user and checks if the trajectory exists on the file system. For spherical trajectories with 900 way-points, this component can save up to 45 minutes of waiting time by persisting trajectories on the file system.

Calibration The calibration component calculates the exact position and orientation of the calibration structure on the acquired image. It uses information about the system, such as the X-ray setup's geometry or the sample holder. The optimal parameters are calculated with the Levenberg-Marquardt non-linear optimization algorithm and the robotic arm's sensor readings as the initial solution.

RobotControl The robot control component interfaces the *MoveIt!* motion planning pipeline. It provides an abstraction layer for the remaining components for critical tasks such as moving the robotic arm to a given pose or stopping it.

ImageSegmentation The image segmentation component processes the acquired images (contrast enhancement) and segments the circles on the calibration structure. We use the circle Hough transform algorithm for segmentation.

ExperimentPreparation The experiment preparation component creates the required directory structure for the requested experiment.

SampleAcquisition The sample acquisition component executes a given trajectory and acquires an image. It interacts with the detector in the laboratory environment to retrieve the image and applying the flat-field correction. In the simulation environment, it requests simulated measurement.

SampleHolderManager The sample holder manager component manages the samples and sample holders available to our system on the file system. The user's sample and sample holder choices are communicated with the remaining components in our system via the sample holder manager. This communication includes updating the coordinate transforms for the sample and sample holder depending on their size. Another example is updating the sample volume for the simulated CT measurements in the simulation environment.

ImageRetriever The image retriever component provides the experiment images to the web user interface. The component fetches the requested image when the experiment identifier and image type are provided. The returned image is resized to 700×700 pixels for reduced network traffic.

VolumeSegmentation The volume segmentation component is only used within the trajectory optimization pipeline. This component is responsible for segmenting the highly absorbing parts of the intermediate reconstructions.

SampleHolderStatePublisher The sample holder state publisher component publishes information about the geometry of the currently mounted sample holder part. The geometry information is published to the *tf* component. Information about the sample holder geometry is critical, as it influences the correct positioning of the sample into the X-ray beam.

CameraStatePublisher The camera state publisher component publishes information about the X-ray acquisition geometry and the position and orientation of the depth cameras. The active collision detection algorithm uses the published information about the depth cameras.

RobotEnvironmentManager The robot environment manager component implements the passive collision detection mechanism. The implementation manages the existing environment of the robotic arm and modifies the arm's configuration space when the user interacts with the user interface. The user can, for example, modify the environment of the robotic arm by choosing a sample holder from the user interface. The existing environment includes the fixed surroundings of the arm and a cubic collision object that models the X-ray beam. The cubic object for the X-ray beam prevents the arm from placing its links into the beam, preventing occlusions in the acquired images.

SphereSampler The sphere sampler component provides an abstract interface for search patterns on a discretized sphere surface. This component is only used within the trajectory optimization pipeline.

SphereListVisualizer The sphere list visualizer component provides an interface for visualizing a set of sphere markers in *RViz*. This component is currently only used by the active collision detection algorithm. The markers are displayed along with the robotic arm and its environment to debug the collision detection algorithm and visualize the currently detected collision.

TrajectoryPlotter The trajectory plotter component can plot circular and spherical trajectories on a spherical surface that is projected into two dimensions with the *Mollweide* projection. In the conventional experiment execution, the trajectory is planned and saved to the file system before it is executed. The trajectory plotter component listens for changes in the corresponding directory and saves a plot of the new trajectory in the same directory. In the online trajectory optimization pipeline, the plotter is called explicitly, and the plot is saved to the corresponding experiment's directory, as the trajectory is unique in this execution mode.

VolumeUtils The volume utilities component implements two methods that provide correct usage of the samples and sample holders in the simulation environment. The first method places spheres inside the holes of the currently selected sample holder for calibration purposes. The second method concatenates multiple volumes into a single volume so that (multiple) samples can be combined with the chosen sample holder for simulating measurements.

RobotStateReset The robot state reset component checks for invalid configuration of the robotic arm's joints directly after the arm is spawned in the simulation environment. The component fixes the invalid configuration by switching the arm's controllers and moving the joints to a predefined valid configuration.

FlatFieldCorrection The flat field correction component applies flat field correction on the acquired images immediately after image acquisition. The correction operation is only required and used in the laboratory environment. The correction operation is applied based on the most recent *empty* and *dark* measurements on the file system.

Gripper The gripper component provides grasp and release functionality for the gripper of the FRANKA EMIKA Panda robotic arm in the laboratory environment.

DetectorManager The detector manager component is responsible for acquiring the required images for the flat field correction in the laboratory environment. The component acquires a predefined number of images and saves their mean to the file system for later use during sample acquisition.

DetectorClient The detector client component receives the raw 16-bit gray-scale image from the workstation directly connected to the detector in the laboratory environment. The image is subsequently converted into a suitable format for communication between ROS nodes and returned to the caller.

RobotStateManager The robot state manager component listens for hardware reflexes of the robotic arm during trajectory execution in the laboratory environment. When it detects a reflex, it initiates the error recovery routine provided by the manufacturer.

4.2.2.2. Utilities

We implement the reusable functionality of our project in the utilities folder. The code in this package is not directly tied to a specific component. In the following, we provide an exhaustive list of the core namespaces and their functionality:

ROSUtils The ROS utilities namespace provides common conversions from foreign types (e.g., *json*, *Eigen*) to ROS-compatible types. This namespace also provides conversions for ROS internal types.

SampleHolderUtils The sample holder utilities namespace provides an abstract interface for reading the sample holder and sample metadata from the file system.

SphericalDataUtils The spherical data utilities namespace provides valuable methods for the trajectory optimization pipeline's score initialization and score update steps.

TrajectoryUtils The trajectory utilities namespace handles trajectory serialization on the file system with the *JSON* data format. Additionally, the namespace provides methods for sphere surface discretization and pose sampling on the discretized surface. These methods are critical for generating spherical trajectories.

Constants The constants namespace defines important constants for the robotic sample holder project components. These constants include all service and action endpoints defined by our components, specific constants for the FRANKA EMIKA Panda robotic arm, string keys for important parameters saved on the ROS parameter server, and miscellaneous other constants.

Configuration The configuration namespace provides an abstract interface for reading configuration parameters from the ROS parameter server. Our components request the parameters by their keys defined in the constants namespace (see sec. 4.2.2.2).

ExperimentUtils The experiment utilities namespace handles experiment serialization on the file system with the *JSON* data format. Additionally, the namespace provides methods for calculating cryptographic hash values of the robotic arm's surroundings (configuration space) for the secure execution of existing trajectories in the arm's current environment.

FilesystemOperations The file-system operations namespace provides an abstract interface for file-system operations to our components and utility namespaces.

MeshUtils The mesh utilities namespace provides methods for reading the mesh files provided by the manufacturer of the robotic arm's links. Information about the mesh structure is converted to suitable data structures for later use in the active collision detection algorithm. The mesh utilities namespace also provides methods for manipulating and preparing the mesh structures.

MoveItUtils The MoveIt utilities namespace provides forward and inverse kinematics methods with the robotic arm in its current environment. Additionally, the namespace implements methods for executing existing trajectories with the arm.

RobotGeometry The robot geometry namespace provides a method for modifying a given pose such that the output pose shifts the goal position from the center of the robotic arm's two fingers to the sample holder's sample plate. This method is essential for precisely placing and intersecting the sample holder and the sample in the central X-ray.

4.2.2.3. Computed tomography

The computed tomography package implements two necessary components to simulate tomographic measurements in the simulation environment.

ComputedTomography The computed tomography component implements the tomographic imaging functionality for the simulation environment. This functionality includes forward and back-projection of samples and image reconstruction. The forward projection relies on a volume that includes the currently used sample holder and sample. This component manages a 3D volume that contains the user's sample and sample holder choice. The component also implements a weight calculation routine for the online trajectory optimization pipeline.

MeshReader The mesh reader component converts 3D mesh structures representing surfaces into 3D volumes. The conversion is handled by the *VTK* library, and the user can specify the desired resolution and spacing for the resulting volume.

4.2.2.4. Safety

The safety package implements the active collision detection algorithm mentioned in section 4.2.3.2. The package also includes components that can create and execute integration tests for the collision detection algorithm within the simulation environment.

CollisionDetector The collision detector component implements the host part of the active collision detection algorithm described in section 4.2.3.2. The responsibilities of the host code include

- subscribing to a depth camera point cloud,
- compiling the device (GPU) OpenCL code,
- subscribing to the current state of the robot,
- subscribing to the trajectory topic of the robotic arm,
- visualizing the detected collision points,

and miscellaneous other tasks. This component will only be compiled and usable if a valid OpenCL runtime (version 2.0) is detected on the system. We describe implementation details for this component in section 4.2.3.2.

OpenCLUtils The OpenCL utilities namespace provides methods for compiling and executing device code with the OpenCL runtime.

TrajectoryRecorder The trajectory recorder component is part of the integration testing functionality for the active collision detection algorithm. This component listens for incoming trajectories and saves the trajectory to the file system for later use in test cases.

CollisionDetectorTest The collision detector test component executes integration tests for the active collision detection algorithm. The component reads trajectories and predefined environments of the robotic arm from the file system and executes the trajectory in the simulation environment.

4.2.2.5. Messages

In ROS, the system architecture requires each component to run in its independent process. Therefore, inter-process procedure calls and data exchange relies on serialized messages sent over the network to predefined topics. The messages package contains all the message definitions of our project.

This package can be subdivided into action, service, and message types. Action types are defined for tasks with expected long execution time (> 5 seconds). Service types, in contrast, are defined for tasks with shorter execution times than actions (< 5 seconds). *Message types* extend the capabilities of simple service and action message definitions with custom data types. Without custom *message types*, only primitive data types can be used for procedure calls.

4.2.2.6. Web

The web package is implemented as a *ReactJS* project and was bootstrapped with the *create-react-app* script. We use the Javascript library *roslib* to communicate with the ROS nodes implemented in our project. The *rosbridge* component acts as a WebSocket. It allows us to receive the service and action messages encoded as JSON messages and serialize them so that the ROS Master can handle the request. We implement two very similar so-called *React components* as a graphical user interface to interact with our experiment ROS nodes: *ExperimentForm.js* interacts with *Experiment.cpp* and *OnlineExperimentForm.js* with *OnlineExperiment.cpp*.

The user can start and resume an interrupted experiment from these web interfaces. Different steps of the pipeline can be run independently, and the progress of each step is displayed on the user interface with a progress bar and the resulting image of the given step (e.g., sample acquisition, circle segmentation, calibration). The detector images for the flat-field correction of the measurements can also be acquired from this interface.

The web user interface works without restrictions in the simulation and the laboratory environment.

4.2.3 Collision Detection

We implement two kinds of collision detection in our system: passive and active collision detection. Both mechanisms aim to provide safe operation of the robotic

arm by the operator. The critical difference between these mechanisms is the point in time when they are invoked.

Passive collision detection is a crucial ingredient to the motion planning problem, as it changes the composition of the configuration space. This modification happens when the trajectory is generated and is therefore done before the robotic arm moves.

The active collision algorithm starts in parallel to the trajectory execution, and the trajectory is passed to this algorithm as input. The process that implements this algorithm can interrupt the robotic arm and the trajectory execution.

In summary, passive collision detection is a proactive measure that helps to generate a safe trajectory for the robotic arm. In contrast, active collision detection actively checks for collisions while the arm moves by analyzing input data from the depth cameras.

In the following, we provide a detailed outline of both methods and point to their location in our project implementation.

4.2.3.1. Passive collision detection

The passive collision detection reads user-defined configuration files containing information about nearby objects from the filesystem at system startup and forwards them to the manipulation framework. The framework passes these objects on to the inverse kinematics, so they are considered obstacles in the *configuration space* when planning the trajectories. We have provided more details about the management of obstacles in the configuration space of the robotic arm in section 4.2.4.

4.2.3.2. Active collision detection

The following paragraphs contain excerpts from the methods section of our first journal publication [Pek+22b], where we also outlined our active collision detection algorithm.

Active collision detection is a process that retrieves the currently executed trajectory and a point cloud from the depth cameras. It then checks if there are potential collisions in the retrieved point cloud with the current trajectory of the robotic arm. If collisions are detected, the arm is stopped immediately.

A similar collision detection mechanism for robotic arms was already implemented in [Her+14]. While the concepts are similar to our implementation, we decided to implement our algorithm for easier integration with our robotic arm and software package.

The process that handles the collision detection loads the 3D structure of the robotic arm's links from the filesystem, which will be used together with the current joint angles of the arm to calculate the approximate position of the robotic arm's links.

The active collision detection mechanism is triggered when a trajectory is received, and the robotic arm starts to move. The trajectory consists of a set of joint angles for each of the seven joints of the robotic arm and is transferred to a GPU together with the point cloud from the depth cameras and the 3D structure information of the robot's links.

We remove all points from the point cloud resembling the robotic arm; otherwise, the robot would be registered as a colliding object. The point removal is done by a self-filter that calculates the distance of each point in the point cloud to individual points on each link of the robotic arm and removes all points from the point cloud that fall below a certain threshold. Afterward, we calculate the arm's movement in 3D space while executing the current trajectory. This movement profile is compared to the current point cloud input from the depth cameras and checked for collisions.

In an additional verification step, the algorithm checks if the reported collision points from the point cloud are not the result of a noisy measurement by checking that the 27 neighbors in discretized 3D space are also collision points. Finally, the points that fulfill this criterion are reported as actual collision points.

If there are collision points after the noise filtering step, we stop the trajectory execution by the robotic arm.

In the remainder of this section, we will outline the specific components of our implementation and their interaction with each other. The explanations and figures provided here were also partly published on my personal blog [Pek].

Implementation. The algorithm consists of various components that fulfill different tasks:

- CollisionDetector: Host / CPU code written in C++ as a ROS node. This node coordinates the communication with the rest of the components. It is also responsible for managing the device / GPU interaction and resources.

- MeshReader: ROS node written in C++. This node is responsible for converting the mesh surface data of the robotic arm's link into 3D volumes.
- SphereVisualizer: ROS node written in C++. This node is solely responsible for visualizing points in the robot's surroundings in RViz.
- detection.cl: Device / GPU code for determining which points in the point cloud are a potential collision.
- verification.cl: Device / GPU code for deciding if a point in the point cloud marked as a collision is a collision or just noise resulting from poor measurement accuracy of the depth camera.

In Fig. 4.9, you can find a sequence diagram showing how the components interact.

In the following, we will discuss the collision detection procedure in detail.

The procedure starts with the init() method in the CollisionDetector ROS node. It consists of the following steps:

- A set of configuration parameters are read from the ROS master
- The private node parameters that were passed to the node at its launch are read into variables
- The robotic arm's mesh surface files are converted into volume files. These volume points are filtered/reduced to the essential points to reduce the algorithm's runtime.
- The kinematics model of the robotic arm is read from the ROS master
- The Denavit-Hartenberg parameters of Panda are set as compile time parameters to the detection OpenCL kernel
- Essential coordinate transforms are retrieved
- Static data is sent to the GPU device by allocating a buffer for each variable. A particular case here is the 3D OpenCL image that discretizes the surroundings of the robotic arm into a 3D volume.
- The robotic arm's link volume points are sent to SphereListVisualizer for visualization in RViz (see Fig. 4.10 (a)). The visualization is helpful because we can check the mesh surfaces for proper conversion and placement into the right coordinate frame in the visualization. Otherwise, the visualization will not show the green markers in the correct shape or position.

joint states callback. The joint states callback captures the current state of the robotic arm. In our implementation, it sends the state of the arm to the GPU whenever an update is received. In the simulation, the joint state topic publishes at a rate of 30 Hz, which means that the GPU receives an update 30 times per second. The update is written asynchronously to a device buffer. However, we should only use the state of the arm before the point cloud was captured. If we use a state of the arm that was captured after the point cloud measurement was received, the filter will try to remove the robotic arm from the point cloud at a position that will be captured in the future. Here, the word future stands for the next frame of the depth camera. The time lag mentioned above means we should store joint states in advance and use the state received just before the point cloud was captured. For this reason, we are using a std::deque and a std::map for storing a set of joint states in the correct order. The deque data structure provides efficient push and pop operations with the First-In-First-Out (FIFO) principle. With this data structure, we can store and delete joint states in the correct order: The state of the arm that first entered the deque will also be the first state that leaves the deque when a specific capacity is reached. The robotic arm's current state is sent to the GPU as actual transforms for each link in 4×4 matrices (rotation and translation).

trajectory callback. The trajectory consists of multiple way-points, each consisting of 7 angles: one angle for each rotational joint of the robotic arm. As we did above with the joint states, we could calculate each link's transform matrix at each way-point. These transform matrices would consume considerable bandwidth on the GPU when accessing the device buffer in the kernel code. We will see later that we have to reduce the amount of data stored in the local memory of the execution units as much as possible. For this reason, we employ the Denavit-Hartenberg parameters to parametrize the robot states on the trajectory way-points. The Denavit-Hartenberg parameters for each link are set as compile-time parameters for the OpenCL kernel except for the parameter theta, which describes the angles of the rotational joints at the way-points is written to the device buffer when the trajectory is received in the respective callback method.

point cloud callback. The point cloud callback is executed every time a point cloud frame is captured and published by the depth camera in the simulation. The callback checks if a trajectory was received beforehand because otherwise, the robot will stand still, and processing the point cloud is unnecessary. If a trajectory was received and written to the GPU, the received point cloud is written to the GPU buffer, followed by a series of data transfer calls. The selection mentioned above of the correct state of the robotic arm is also handled in this callback. After pushing all of the necessary arguments to the GPU, the computation on the device is started by

launching a thread (also called a work item) for each point in the point cloud. The first kernel running on the GPU is the detection.cl kernel. This kernel decides which points in the point cloud are a potential collision. In order to do that reliably, the robot's surroundings are discretized in a 3D volume. The discretization means that every measurement in meters near the robotic arm is mapped to a voxel in a volume with predefined dimensions (in our case, $512 \times 512 \times 512$). When a point in the point cloud is reported as a potential collision for the currently executed trajectory, we determine its coordinates in the aforementioned discretized volume. This volume coordinate is marked as a potential collision candidate and will be further checked later. This kind of discretization has the advantage that no magic threshold for the number of potential collision points has to be selected, after which we can state that a collision would occur. There might be noisy measurements in the point cloud, after all. This threshold would be set to distinguish between sensor noise and a definitive collision point.

The second kernel is invoked after receiving the results from the first one. It applies simple neighborhood checking on the discretized volume and returns a definitive statement on whether a collision is present on the trajectory. The number of collision points is printed on the console. In case of a collision, a stop message is sent to the robot control node to interrupt the trajectory execution. The collision points are converted from the discretized volume coordinates to their original coordinate frame with a meter representation. We send these coordinates to the SphereVisualizer node, which publishes them to RViz as 3D markers. The user can see the location of the collision points in RViz.

detection kernel. The detection kernel has two tasks: Filtering the robotic arm from the given point cloud and checking the remaining points in the point cloud for a collision with the trajectory that is currently being executed.

robot self-filter. In order to filter the robotic arm from the given point cloud, we check the distance of a point from the point cloud to each of the volume points of the robotic arm. The volume points were transferred to the GPU in the point cloud callback. If the distance of the point falls below a given threshold (10 cm) for all of the link volume points, we exit the kernel without further computation. The volume points are placed at the correct positions using the link transforms that were transferred to the GPU whenever a new update of the robotic arm's joints was received from the simulation.

collision detection. If the point does not lie within the robotic arm's estimated position, we check if it lies on the trajectory the robotic arm is executing. This collision check is done with the following steps:

1. Loop over all trajectory way-points
2. For every point in the robotic arm's volume, determine its position in the current and the next way-point of the trajectory
3. Check the distance of the given point from the point cloud to the volume point's location on the current way-point
4. Check the distance of the given point from the point cloud to the volume point's location on the next way-point
5. Check the distance of the given point from the point cloud to a line that connects the volume point's location on the current and next way-point
6. If one of these three distances is below a predefined threshold (3 cm), we assume a collision

The third collision check (5.) constructs a cylinder from the volume point's current and next location on the trajectory. This check ensures that the point from the point cloud does not lie within this cylinder. We estimate the motion of this part of the robotic arm in a given time frame as a linear motion from one point to the other.

If a collision is detected, the coordinates of the given point from the point cloud are converted to its index in a predefined discretized volume. This volume encompasses the robotic arm's reachable area. We maintain the discretized volume as an OpenCL 3D image on the device; hence, we set the value at the index we calculated earlier to 1 to indicate that this part of the volume contains a collision.

verification kernel. The verification kernel ensures that the reported collisions in the discretized volume are not just noisy measurements but actual collisions. If and only if all neighbors of a given collision voxel are also collision voxels, it confirms that there is a collision. The result is written to a flat array with the discretized volume's size.

Performance evaluation. The collision nodes can be started with the visualize flag to get more debugging information on the console and in RViz. When this feature is enabled, the collision detection runs slower as more data is transmitted between the CPU and the GPU for visualization purposes. Each frame of the point cloud is processed in 150 milliseconds on average. When the visualization is disabled, we have an average processing time of 35 milliseconds per camera frame. These numbers were measured on a workstation with a compatible AMD Radeon VII GPU running two independent collision detection processes simultaneously, one per depth camera.

4.2.4 Path Planning

With the path planning procedure, our system exposes an abstract interface to the user for planning and executing advanced trajectories. A detailed description of the path planning process can be found in our publications [Pek+22b; Pek+23b] and sections 5.2.2 and 6.2.2. In section 5.2.2, we describe the fundamental problem of path planning with the robotic sample holder and our solution approach to centering the sample's center at the central X-ray. In section 6.2.2, we describe our approach to planning and executing more complex trajectories, including the ability to recover from hardware reflexes that can arise in practice.

In this section, we will briefly overview two components in our software package and their implementation details that are critical to our path planning pipeline.

The passive collision detection mechanism is a vital component of the MoveIt path planning pipeline. We have implemented an abstraction layer for managing the environment of the robotic arm, the *RobotEnvironmentManager* component described in section 4.2.2.1. It reads the potential collision objects of the robotic arm from the file system and adds them to the so-called *planning_scene*, a topic that MoveIt maintains for updates in the robot's environment. Our component also offers an interface for adding and removing samples and sample holders to the planning scene, which is needed because the user can swap samples and sample holders at runtime through the user interface. Additionally, our component adds a cubic collision object for the X-ray beam to the planning scene and manages which samples, sample holders, and the robotic arm's links are allowed to enter the field of view of the beam. We have visualized an exemplary collision environment in Fig. 4.11.

The coordinate transformations for the samples and sample holders are other vital ingredients to the motion planning pipeline. We manage these transformations with the *SampleHolderStatePublisher* component described in section 4.2.2.1. When the user selects a new sample or sample holder from the user interface (see Fig. 4.8), the sample holder state publisher component is updated. The *SampleHolderManager* component validates the user's choice in an intermediate step. The connection to the motion planning pipeline occurs when the user requests a trajectory: The coordinate frame for the trajectory way-points' goal poses is the sample's center. Every sample has different dimensions and can be attached to one of many sample holders that differ in dimensions and shape. The SampleHolderStatePublisher component is responsible for publishing the coordinate transformations for the selected sample with the correct position and orientation to the *tf* component of the core ROS infrastructure.

4.2.5 Software Stack

The central part of our software stack is the *Robot Operating System* (*ROS*) [Qui+09], which is a middleware for the communication of independent processes across a network. We accomplish robot manipulation with the *MoveIt!* framework [Col+14; SC] and the *franka_ros* configuration package [FRA21]. For image processing tasks and the circle segmentation, we use *OpenCV* [Bra00], for multithreading on the CPU *OpenMP* [DM98] and on the GPU *OpenCL* [SGS10] and for the tomographic reconstruction *elsa* [LHF19]. The scientific calculations in section 6.2.4 are implemented with *scipy* [Vir+20]. We read the 3D mesh files of the robotic arm with OpenMesh [Bot+02]. The sphere discretization in section 6.2.3 was implemented with the HEALPix C++ and Python interfaces [Gor+05; Zon+19]. For calculating sha256 sums of the configuration spaces mentioned in section 6.2.2, we use crypto++ [@Dai22]. The 2D reachability maps in section 6.3.1 were generated using matplotlib and basemap [Hun07].

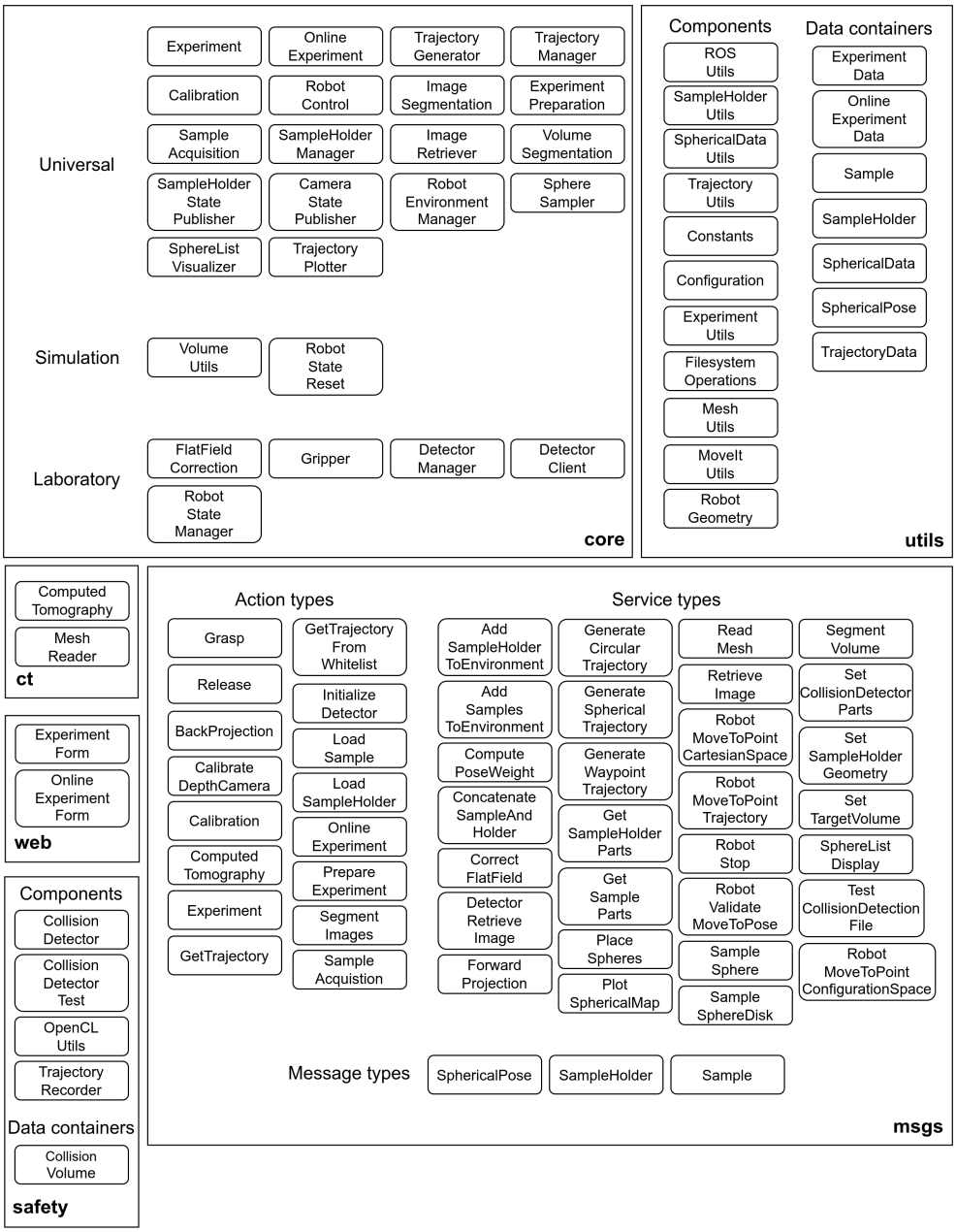


Fig. 4.7.: System architecture. We illustrate the system architecture and different components within the sub-folders of our software project. The packages **core**, **ct**, **web**, and **safety** contain code that interacts directly with the *ROS master*. The **utils** package provides important utility classes and methods. The **msgs** package defines common message types for the interaction of the components with each other.

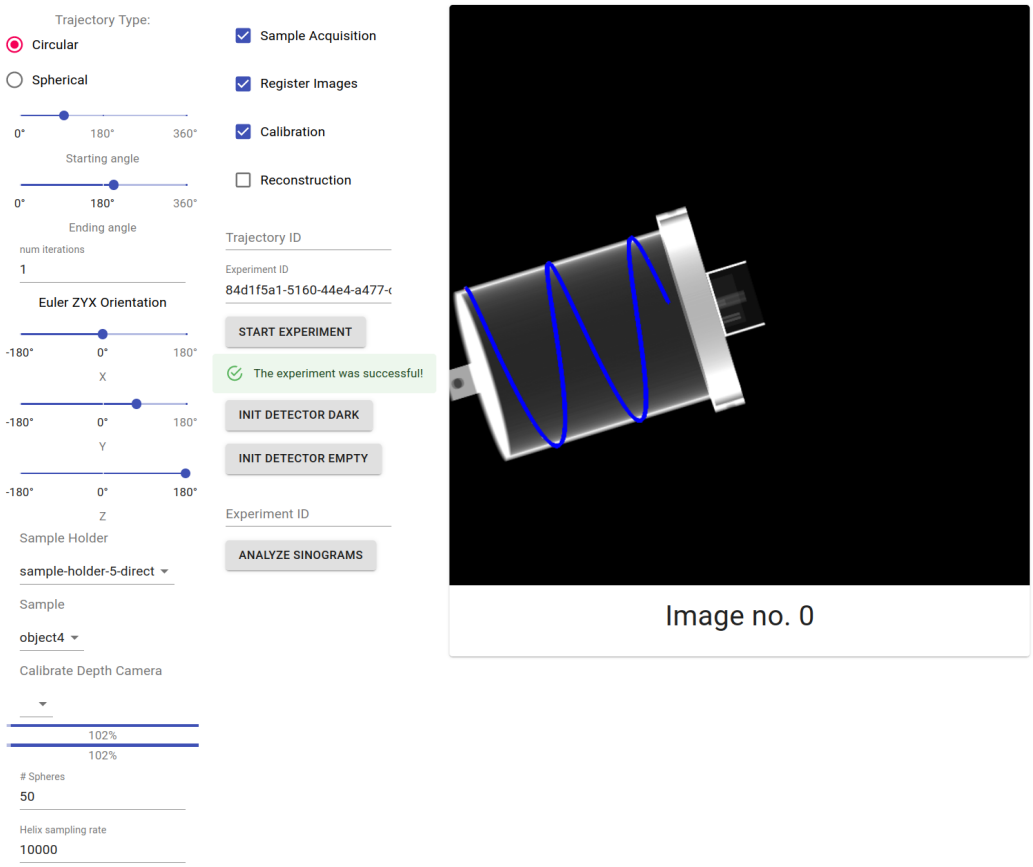


Fig. 4.8.: Web user interface for the robotic sample holder. The user interface is accessible from the web browser in the simulation and laboratory environment and provides controls for defining essential experiment parameters like trajectory type. The interface also shows intermediate measurement results.

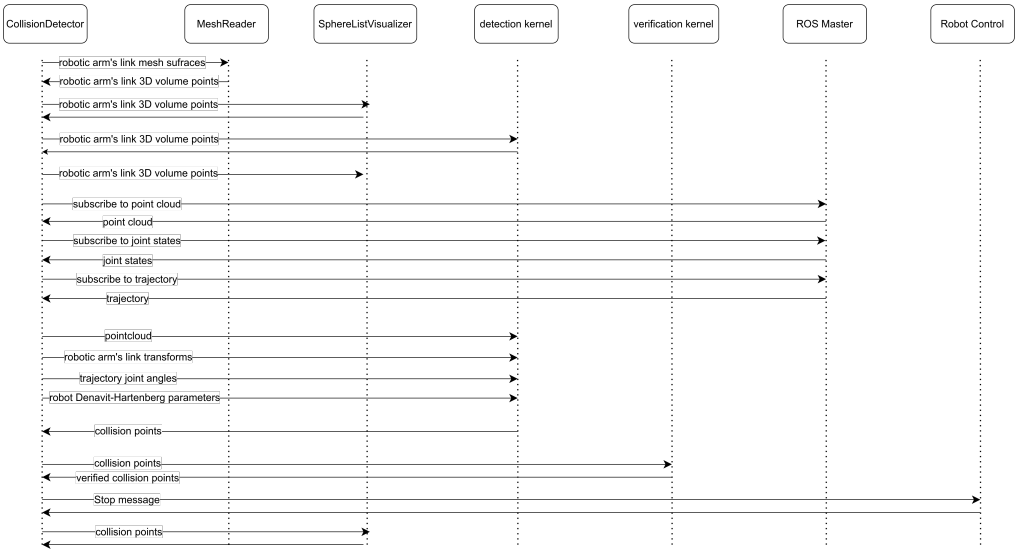
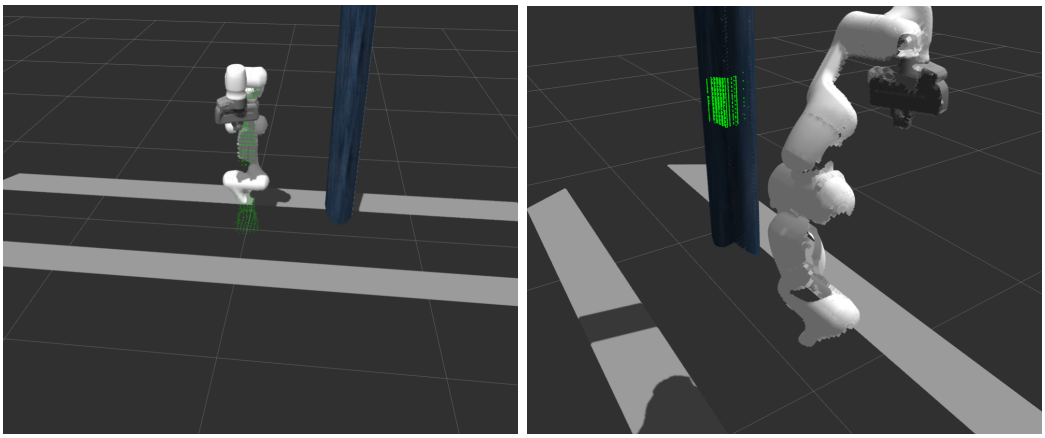
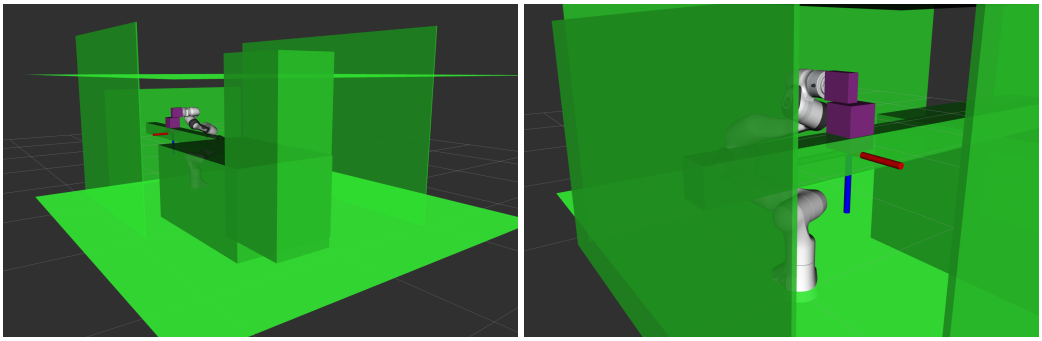


Fig. 4.9.: Active collision detection. Sequence diagram.[Pek] The sequence diagram illustrates the active collision detection components' interaction.



(a) Pointcloud initialization. (b) Collision points.

Fig. 4.10.: Active collision detection. Visualizations. [Pek] In (a), our active collision detection algorithm's internal representation of the robotic arm's current state is visualized. This visualization is a debugging feature, and it is essential for verifying that the arm's 3D mesh files were read in and fused with current joint angles correctly. In (b), we visualized an exemplary collision of the robotic arm with a telephone pole in the simulation environment.



(a) Robot environment far (b) Robot environment close up

Fig. 4.11.: Passive collision detection. Visualizations. Figures (a) and (b) depict the objects that are encoded in the robotic arm's configuration space for the passive collision detection mechanism described in sections 4.2.3.1 and 4.2.4. Our software package places the green objects at system startup while it swaps the violet objects at runtime, depending on the user's choices.

5 X-ray Computed Tomography with seven degree of freedom Robotic Sample Holder

5.1 Introduction

In chapters 1 and 4 of this dissertation thesis, we have pointed out that industrial robotic arms' repeated placement accuracy is insufficient for tomographic imaging purposes. We have identified that a geometric calibration procedure is necessary for determining the exact position and orientation of the sample on the detector images. In this chapter, we will present our study on the repeated placement accuracy of the FRANKA EMIKA Panda robotic arm and our calibration algorithm for identifying the exact position and orientation of the sample. We will introduce an intermediate sample holder part that will serve as a calibration target between the arm and the sample.

This chapter is an adapted version of a peer-reviewed manuscript [Pek+22b] and a conference publication [Pek+22a] published during my Ph.D.

In the following sections, we present an X-ray Computed Tomography setup that integrates a seven degrees of freedom robotic arm as a sample holder within an existing laboratory X-ray computed tomography setup. We aim to provide a flexible sample holder that is able to execute non-standard and task-specific trajectories for complex samples. The robotic arm is integrated with a unified software package that allows for path planning, collision detection, geometric calibration and reconstruction of the sample. The calibration is necessary to identify the accurate pose of the sample which deviates from the expected pose due to inaccurate placement of the robotic arm. With our software the user is able to command the robotic arm to execute arbitrary trajectories for a given sample in a safe manner and output its reconstruction to the user. We present experimental results with a circular trajectory where the robotic sample holder achieves identical visual quality compared to a conventional sample holder.

The system can easily execute specific trajectories that can overcome the limitations of fixed trajectories. Arbitrary rotations can be reached with the robotic arm's seven degrees of freedom (seven rotational joints). This will enable imaging modalities that require non-standard acquisition sequences in the future, such as Anisotropic X-ray Dark-field Tomography (AXDT). AXDT is a novel imaging technique that allows the extraction of X-ray scattering and phase contrast information by employing grating interferometers [Wie+16; Wie+18]. The robotic sample holder will enable arbitrary rotations covering the full sphere and hence expose the 3D structures of the target object by measuring the full dark-field contrast from all possible angles. Conventional sample holders pose a significant challenge when the acquisition trajectory is required to cover the full sphere of rotations, as more intervention by the user is required and it may not be possible to measure the sample from certain angles.

In the remainder of this section, we will provide an overview of related work on imaging with robotic arms and geometric calibration mechanisms for X-ray CT.

5.1.1 Related work on X-ray CT with Robotic arms

Robotic arms were also used in the past in computed tomography systems [Zie+20; LHH19; @Hea21]. The main difference to our work is the kind of robotic arm that is used. It offers a higher flexibility than the robotic arms that were used in related work due to its seven degrees of freedom and it has two fingers that make chained pick-and-place tasks possible without user intervention.

In [Zie+20] the source and detector are mounted on robotic arms and the sample is centered between the source and the detector by the arms. The main difference to our work is that this is not a laboratory scale setup but assembly line scale and that the sample does not move but the source and detector. Therefore it is not directly comparable to our setup. The detector pixel size ($100\mu m$ theirs vs $150\mu m$ ours) and the voxel resolution of the reconstructions ($70\mu m$ vs $100\mu m$) is very similar to our work. The authors do not specify the exact models of the robotic arms, but from the figure it can be seen that they have four degrees of freedom (compared to seven with our robotic arm). The reduced degrees of freedom count means the robot is less flexible and it has difficulties reaching certain acquisition angles.

In [LHH19] the authors demonstrate the advantage of non-circular CT scanning trajectories. The experiments are conducted in a simulation using a 3D model of the specimen. With a circular scanning trajectory the specimen absorbs X-rays

when spheres are added to it and reconstruction quality is impacted resulting in streak artifacts. It is demonstrated that by using a simulated six DoF robotic arm a simulated non-circular trajectory that almost covers the full rotational sphere would be possible and result in a reconstruction with no artifacts.

The Siemens Healthineers *Artis zeego eco* angiography platform [@Hea21] consists of a single five DoF robotic arm which is moving the detector and source with a fixed distance between each other. The main differences to our work are that the detector and source are both mounted to the robotic arm and hence are moving parts. Also the sample in this case is a living patient. The robotic arm is positioned such that the body part of the patient which is of interest lies exactly between the source and detector.

In [Her+21], the authors present an X-ray tomography system with two robotic arms. The X-ray source and the detector can be moved independently from each other by the two arms, and the sample is mounted statically between them. There are two key differences to the system that we propose. The first is that we are moving the sample, and thus our system only requires one robotic arm instead of two. Furthermore, moving the sample does not restrict the system to movable X-ray sources and detectors; hence, it is more flexible. On the other hand, this means that we restrict the samples in our system in size and weight, and the sample should not deform or otherwise change when moved. The second key difference is that our robotic arm is significantly smaller and thus fits into an existing X-ray CT setup. In contrast, the robotic arms used in [Her+21] can reach up to 3 meters of height (compared to 1.2 meters with our arm), which might not even fit into an existing laboratory [@KUK21; @EMI20]. Furthermore, the smaller robotic arm in our system is more affordable. The most significant technical differences are that our robot has an additional degree of freedom (ours seven vs. theirs six) but worse repeatability when the identical trajectory is executed repeatedly (ours 0.1 mm vs. theirs 0.04 mm). The higher degree number increases the probability of our system reaching positions on complex trajectories. In contrast, our system tackles its more inaccurate repeatability with the calibration procedure described in this chapter.

5.1.2 Related work on geometric calibration in X-ray CT

The seven joints of our robotic arm imply seven degrees of freedom but a higher number of joints also introduces a higher error on the placement of the sample. A calibration mechanism is needed for determining the projection parameters of the sample. The projection parameters can be split into the external and internal camera

parameters. The external parameters are the rotation and position of the camera relative to the sample (or vice versa). The internal parameters concern the camera system itself. In our imaging system the internal parameters are fixed and they are determined beforehand.

In [LZL10] the authors propose a generic calibration method for tomographic imaging systems with flat-panel detectors. A flat calibration phantom with 44 spheres in total on two parallel planes is used for calibration. Both sets of camera parameters (internal and external) are extracted from the images in a direct computation step.

In [Cho+05] the authors propose a calibration method based on a cylindrical calibration phantom similar to what we will use here. At least two sets of spheres in a circular arrangement are needed in order for this approach to work because this allows the extraction of the center of the calibration phantom's coordinate system. The geometric parameters are computed directly with a closed analytical expression for each image. The parametrization of the geometry only permits rotational acquisition trajectories, whereas our calibration method works with arbitrary placements of the sample and hence any kind of trajectory.

In [Rob+09] the authors propose a method similar to [Cho+05]. A calibration phantom with two sets of spheres that are arranged in an ellipse is used. The difference to [Cho+05] is that the geometric parameters are not calculated directly but an optimization step is introduced for computing the rotation parameters. Additionally, this method is valid for arbitrary geometries, not just rotational trajectories compared to [Cho+05]. The main difference to our method is that two ellipses are used for calibration instead of a helix.

In summary when compared to [LZL10] our method offers more flexibility in terms of the constraints on the calibration phantom because we do not restrict the arrangement of the markers to obtain a certain 3D coverage. When compared to [Cho+05] and [Rob+09] our approach is much simpler in terms of the analytical expressions that are needed and it is decoupled from the specific geometrical structure on the calibration phantom. And in general, our approach is more suitable for robotic arms with higher degrees of freedom because it is valid for arbitrary geometries as opposed to the methods in [LZL10] and [Cho+05] which are restricted to rotational acquisition trajectories.

5.2 Methods

In this section the methods for operating the robotic arm as a sample holder in a lab X-ray CT setup are discussed in detail. After introducing the hardware components and software architecture of the system more specific parts like path planning, collision detection, calibration and reconstruction are described.

5.2.1 Sample Holder

The sample holder is a critical component of the system as it allows the robotic arm to grasp samples of arbitrary shape and is a fundamental part of the calibration process where the position and orientation of the sample is identified. The 3D models of the sample holder and the rail component are visualized in figure 5.1. The sample holder consists of two parts. The bottom part is where the robot's fingers can grasp the holder steadily. The upper part fulfills the actual purpose of placing a geometric structure around the sample on a cylinder. Prior to attaching the sample holder to the robotic arm's fingers the sample needs to be glued to the mounting plate which is inserted into the cylinder from the top at the intended position. There is no need to screw the mounting plate, as there is enough friction with the cylinder to hold the plate in place (see fig. 5.1).

The cylinder is 5.6 cm tall and 3.5 cm in diameter inside. The sample holder was designed with a 3D modeling software and printed using a 3D printer with accuracy of 0.08 to 0.2 mm on all three axes. The printing accuracy is important as the local coordinates of the spheres in the 3D model are used as reference points in the calibration algorithm.

The geometric structure embedded in the sample holder is a helix which is made up of 50 embedded aluminium spheres of 0.678 mm diameter. These spheres were fixed by hand on notches that were included in the design process of the holder. The spheres appear as circles on the detector images that will be segmented during calibration.

The helix can be parametrized by the following 3D parametric curve:

$$h(\tau) = \begin{pmatrix} u(\tau) \\ v(\tau) \\ w(\tau) \end{pmatrix} = \begin{pmatrix} r * cos(\rho * \tau + \phi) \\ r * sin(\rho * \tau + \phi) \\ \tau \end{pmatrix} \qquad (5.1)$$

$$\tau, r, \rho, \phi \in \mathbb{R}$$

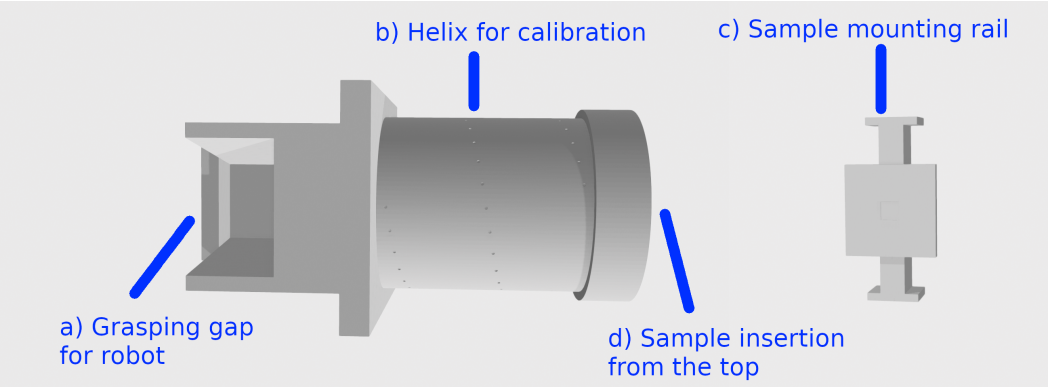


Fig. 5.1.: Sample holder. a) The robot can grasp the sample holder with its two fingers by sliding into a conically shaped gap for easier engagement of the fingers. b) The middle part of the holder houses a helix structure that is used in the calibration step. The aluminium spheres are glued into holes and can then be segmented in the acquired images. c) The sample is fixed on the mounting rail which is inserted from the top of the holder into the hollow cylindrical structure (d).[Pek+22b]

τ runs between the local w coordinates of the first sphere and the last sphere of the helix: $w_{min} < \tau < w_{max}$ where $w_{min}, w_{max} \in \mathbb{R}$.

The parameters r (radius), ρ (frequency) and ϕ (phase shift) parametrize the helix. They can be determined by fitting the sphere coordinates from the 3D model of the sample holder to eq. (5.1) with a least-squares term. The source code of this process can be found in the file *helix_fitter.py* in our repository [Pek21].

The helix can be discretized by choosing a fixed number $H \in \mathbb{N}$ of points $\{\tau_i\}_{i=1,\dots,H} \in [w_{min}, w_{max}]$ for the free parameter τ:

$$h_i = \begin{pmatrix} u(\tau_i) & v(\tau_i) & w(\tau_i) \end{pmatrix}^\top . \tag{5.2}$$

5.2.2 Path Planning

In this section, we are going to outline the important aspects of controlling the robot for use as a sample holder in X-ray CT setups. The main component is task planning which includes all steps that are necessary to place the sample at the correct place. We have provided additional implementation details in section 4.2.4.

Task planning consists of multiple steps. The first step is determining the acquisition trajectory for the given tomographic task. The acquisition trajectory consists of a set of N poses (positions and orientations) for the center of the sample holder that is currently attached to the robot. The acquisition poses are expected to lie on the

intersection of the central ray of the X-ray source with the vertical operating plane of the robotic arm in order to fit the sample (holder) in the limited field-of-view of the imaging system. The poses are targeted at the center of the sample holder because our goal is to image the sample that is contained inside the sample holder. These poses can be generated from the user interface of our software package.

In the second step of task planning information on which particular link of the robotic arm should reach the given pose is added. The kinematic representation (a chain) of the robotic arm is extended by a virtual link that starts at the arm's last link and points to the center of the current sample holder. The path planning algorithm can now place the tail of this virtual link at the goal position. This will ensure that the goal pose, which serves as input to the next step, is exactly at the center of the current sample holder. Without this modification to the kinematic chain instead of the sample holder's center the robotic arm's last link would be placed at the goal pose.

In the third and last step, the inverse kinematics for the given poses are calculated. It calculates the angles at the joints that are required to reach the given goal pose and calculates a series of angles from the given starting position to reach the goal. The resulting inverse kinematics is a series of angles and timestamps (also called *trajectory* in the robotics literature, different than the *acquisition trajectory* from step one) where the first set of angles matches the current state of the robot.

We swapped the default inverse kinematics backend for this task in the *franka_ros* package provided by the manufacturer with the *TRAC-IK* library which has a higher solving rate and a shorter runtime for inverse kinematics tasks on robotic arms with high degrees of freedom like ours [BA15].

5.2.3 Calibration

The calibration procedure tackles the issue that the robotic arm does not sufficiently accurately place the sample at the desired position due to inaccurate path planning and inaccurate electrical motors at its joints. Reading the sensors of the robotic arm and deducing the samples current position is also insufficient to determine the correct position as inaccurate values are reported. However the exact position of the sample at each view is required for the reconstruction. With the calibration procedure we are able to identify the actual positions and orientations of the sample for the reconstruction step. For the calibration a sample holder with an embedded geometric structure that can be detected on the detector images is necessary. A suitable sample holder was introduced in section 5.2.1.

The calibration is implemented in multiple steps (see fig. 5.2). The first step is the post-processing of the detector image. Its contrast is enhanced and a median filter with kernel size 5 is applied to reduce noise and improve the segmentation results. The calibration circles on the image are detected in the next step with the circle Hough transform algorithm [Bal81]. The result is a set of 2D circle center coordinates $\hat{m}_j = \begin{pmatrix} d_{x,j} & d_{y,j} \end{pmatrix}^\top$ on the detector.

Eq. (5.1) and the current position of the robotic arm are now used to project a set of helix points h_i (eq. (5.2)) onto the detector image for comparison with the segmented points $\hat{m}_j$ and determining the geometry of the sample.

For this projection the intrinsic camera matrix K and the external parameters R and t are needed. K is fixed for the current X-ray CT setup and R, t are determined by the robotic arm's current position.

In the following we will provide an overview of the equations that will lead to the final least-squares term that includes the aforementioned comparison algorithm. This least-squares term is used for determining the actual pose of the sample by utilizing an optimisation algorithm.

There are three critical coordinate systems in our setup (see fig. 4.1a). The first is fixed to the X-ray source with x, y and z-axis. The second is fixed to the center of the sample holder with u,v and w-axis and moves with the robotic arm as it is attached to the arm's fingers. The third is fixed to the detector with d_x and d_y axis.

The rotation R of the sample holder relative to the source can be parametrized w.l.o.g. by consecutive rotations about the z, y and x-axis:

$$R(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_x(\gamma) \tag{5.3}$$

t is the offset of the source center to the sample holder's center:

$$t = \begin{pmatrix} x & y & z \end{pmatrix} \tag{5.4}$$

K is fixed and can be calculated with the parameters sdd (source to detector distance), $d_{x,p}, d_{y,p}$ (principal points on d_x and d_y-axis) and d_w, d_h (detector pixel width and height):

$$K = \begin{pmatrix} \frac{sdd}{d_w} & 0 & d_{x,p} \\ 0 & \frac{sdd}{d_h} & d_{y,p} \\ 0 & 0 & 1 \end{pmatrix} \tag{5.5}$$

These parameters are fixed for the current setup and can be determined beforehand.

We introduce the short notation $\zeta = (\alpha, \beta, \gamma, x, y, z)$ for the free parameters. The camera projection matrix P can now be calculated:

$$P(\zeta) = K \begin{pmatrix} R(\alpha, \beta, \gamma) & | & t \end{pmatrix} \in \mathbb{R}^{3\times4}. \tag{5.6}$$

The projection matrix is now used to project a set of $H \in \mathbb{N}$ fixed points $h_i \in \mathbb{R}^4$ on the discretized helix from eq. 5.2 onto the detector:

$$\begin{pmatrix} d'_{x,i}(\zeta) \\ d'_{y,i}(\zeta) \\ d'_{z,i}(\zeta) \end{pmatrix} = P(\zeta) h_i \tag{5.7}$$

$$m_i(\zeta) = \begin{pmatrix} d_{x,i}(\zeta) \\ d_{y,i}(\zeta) \end{pmatrix} = \begin{pmatrix} d'_{x,i}(\zeta)/d'_{z,i}(\zeta) \\ d'_{y,i}(\zeta)/d'_{z,i}(\zeta) \end{pmatrix} \tag{5.8}$$

d'_i are the homogeneous detector pixel coordinates and m_i are the projected analytical helix points on the detector. These points resemble the expected position of the helix structure and they will be used for constructing an error term in the 2D detector image domain.

An appropriate cost function for comparing the error between the current and expected position of a measured circle center $\hat{m}_j$ and a projected point on the helix m_i is the *reprojection error*:

$$E(\zeta, \hat{m}_j, m_i) = \hat{m}_j - m_i(\zeta) \in \mathbb{R}^2 \tag{5.9}$$

Eq. (5.9) will only measure the error for a specific pair of points. In our case there are

- c (detected) circles on the current image,
- s spheres glued onto the holder and

- H projected points on the helix from eq. (5.1).

It is important to note that $c \leq s$ because the segmentation algorithm might fail to detect all circles.

We now compare each of the c detected circle centers $\hat{m}_j$ to all H sampled and projected points m_i and choose the pair with the smallest distance (see algorithm 1).

Algorithm 1 Calibration algorithm: cost function

Require: $\zeta_{step}, circles, helix\_points$
Ensure: residuals_min

$residuals\_min \leftarrow []$
for $j \leftarrow 1$ to $size(circles)$ **do**
  $\hat{m}_j \leftarrow circles[j]$
  $residuals \leftarrow []$

  for $i \leftarrow 1$ to $size(helix\_points)$ **do**
    $m_i \leftarrow helix\_points[i]$
    $residuals[i] \leftarrow E(\zeta_{step}, \hat{m}_j, m_i)$
  end for

  $residuals\_min[j] \leftarrow min(residuals)$
end for

We can formulate this algorithm as a least-squares problem:

$$\arg\min_{\zeta=(\alpha,\beta,\gamma,x,y,z)} \sum_{j=1}^{c} \min_{1\leq i\leq H} E(\zeta, \hat{m}_j, m_i) \tag{5.10}$$

The optimization problem is nonlinear due to the sine and cosine terms in the rotation parametrization. In our implementation we use the *Levenberg Marquardt* algorithm. The *Jacobian* matrix with the partial derivatives of the cost function with respect to the free geometry parameters is not computed directly, but the *2-point* finite difference scheme is used for numerical estimation.

The resulting parameters $\alpha, \beta, \gamma, x, y, z$ can be used for the reconstruction as the geometry of the given acquisition.

5.2.4 Reconstruction

For tomographic reconstruction, the sinogram contained 1000 equidistant X-ray projections along a circular trajectory sized 720×720 pixels with a spacing of $600\mu m$.

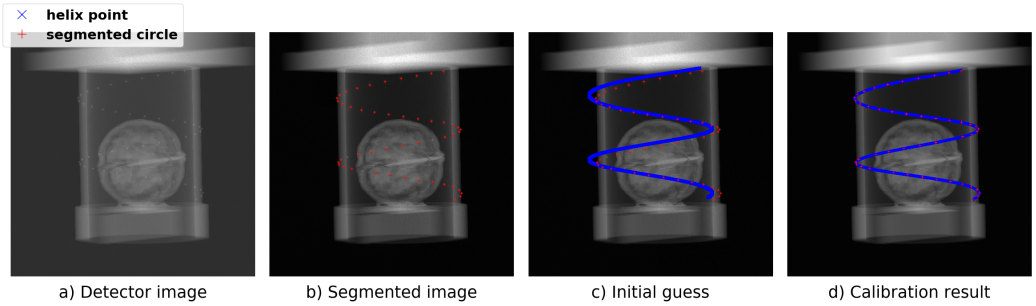


Fig. 5.2.: Calibration procedure. In a) the flat-field corrected detector image is displayed. This image is contrast-enhanced and subsequently a circle detection algorithm is executed. The resulting image where the detected circle centers are marked with red crosses is displayed in b). Given the geometry of the sample holder and the robot's sensor readings when acquiring the image, an initial guess of the helix location (blue crosses) is projected onto the image plane (c). The parameters that define the rotation and translation of the helix are optimized in a least-squares problem in the 2D image domain. The resulting parameters are used to project the helix again to the image domain to display the final outcome of the calibration (d).[Pek+22b]

The reconstruction volume was sized $720 \times 720 \times 720$ with isotropic voxel spacing of $100\mu m$. The remaining parameters of the reconstruction were as mentioned in section 3.3.

5.3 Experiments and Results

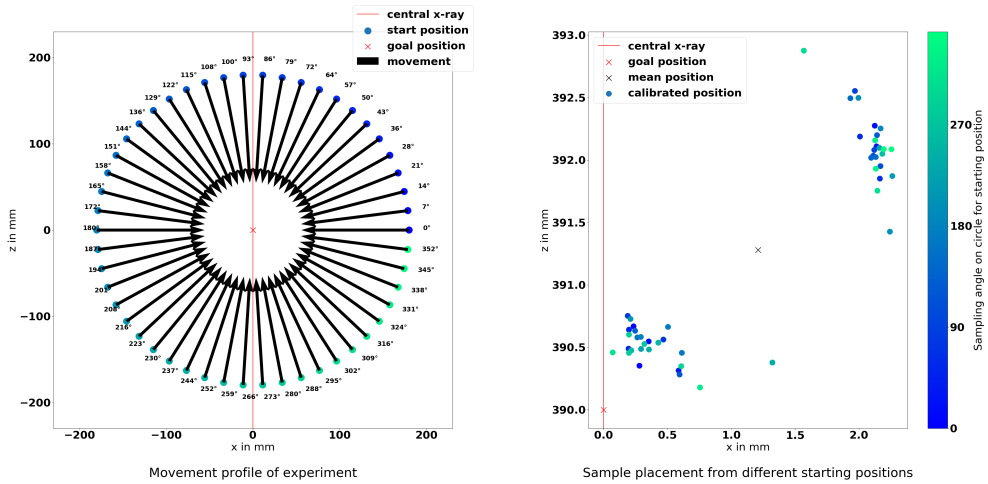
Before running the main experiments, an experiment for determining the accuracy of the robotic arm was conducted. The sample holder from section 5.2.1 was used for all experiments for the purpose of geometric calibration. The collision detection algorithm was running in the background throughout these experiments.

5.3.1 Robot placement error

The robotic arm does not accurately place the sample at the desired goal position and it is also not able to report the current position of the sample accurately. This hinders the reconstruction of the sample. In the following we quantitatively demonstrated the need for a calibration mechanism.

We moved the robot from different starting positions to a predefined identical target pose. The starting positions were sampled on a horizontal circle with radius 180 mm and the center at our default acquisition goal position along the central

X-ray (see fig. 5.3a). The resulting positions were determined by running the calibration algorithm on the acquired images. Deviations from the desired goal position therefore demonstrate the need for a calibration mechanism.



(a) (b)

Fig. 5.3.: Robot calibration experiment. a) Robot placement precision from different starting positions from the top perspective. Starting positions sampled on a circle parallel to the base of the robot with radius of 18 cm and the default goal position for the sample holder as center point. The goal position (cross) and orientation was identical for all measurements. The resulting images were calibrated with the calibration procedure in fig. 5.2. b) The centers of the calibrated helix structures are plotted with circles. The z-axis is omitted for illustration purposes. The expected location of the circles is outlined by the red cross, which lies on the center ray of the source. The distance of the actual position of the calibration structure (circles) to the central ray (red cross) demonstrates that a calibration procedure is needed.[Pek+22b]

5.3.2 Calibration parameters

In this section we are going to state our choices for the three parameters of the calibration method in section (5.2.3):

- a rejection threshold on the reprojection error
- the number of spheres on the holder
- the sampling rate on the helix structure.

The threshold on the reprojection error determines when a calibration result for a given image is not accepted as valid and hence not used for the reconstruction of the

sample. We chose 2.5 pixels distance between each detected circle $\hat{m}_j$ to smallest m_i as threshold. With this threshold the rejection rate of the calibration algorithm was 0.8% with 8 out of 1000 for the example walnut dataset (with $H = 10.000$). All rejections originated from false positives in the segmentation step when random points on the image were detected as circles.

The remaining parameters for the calibration algorithm are the number of spheres and the sampling rate on the helix structure. We have altered one of these parameters while fixing the other one for our walnut dataset and analysed the change in the reprojection error. Our choice for the helix sampling rate ranged from 500 to 30.000. For the number of spheres we sampled a different set of n random spheres from the detected spheres without replacement for each individual image, where $10 \leq n \leq 50$. The results are plotted in fig. 5.4.

For our experiments we fixed H to 10.000.

5.3.3 CT measurements

We conducted four experiments in total: two samples (walnut and pistachio) were each measured with the robotic arm and a conventional rotational stage.

For each CT measurement, 1000 images were acquired with a source voltage of 30 kV, source power of 1445μA, and exposure time of 1s.

In fig. 5.5 a) the reconstruction of the walnut with the rotational stage is compared to the robotic arm as the sample holder. The two volumes were registered manually as we found automatic registration of the two discretized volumes to be unreliable. The slices were chosen manually for illustration purposes. The center slices of the volume from the top and the front view were extracted and cropped to the region of interest.

5.4 Discussion

5.4.1 Robot placement error

We can see in fig. 5.3b that there are two issues: The mean of all points is shifted by more than 1 mm in both directions and the final positions vary significantly from the mean position.

The goal position lies on the intersection of the central ray of the X-ray source with the vertical operating plane of the robotic arm. This means that the goal position depends on the relative distance between the X-ray source and the robotic arm's base. From the shifted mean in figure 5.3b we can conclude that our assumption of the goal position is not correct. This systematic error likely is the result of an inaccurate measure of the relative distance between the source and robotic arm.

The second issue arises from the inaccuracy of the inverse kinematics and the limitations of the electrical motors at the joints. In the ideal case all of the colored points would lie on the same spot. However, the final positions for all measurements form two clusters, without a correlation to the starting point. From the random distribution of the colored points we cannot determine a systematic pattern and hence no clear reason for these positions on the graph can be deduced for the starting angles. There is also no obvious reason for the partitioning of the points in two clusters. From these observations we conclude that a calibration procedure is necessary for determining the position and orientation of the sample and subsequently reconstructing the sample accurately.

5.4.2 Calibration parameters

In section 5.3.2 we have stated our choices for the calibration parameters used in section 5.2.3. For the rejection threshold on the reprojection error of the calibrated images we observed in fig. 5.4b that the lowest mean error per circle is 1.77. This value could serve as a lower bound for the threshold that we are trying to determine because for the given number of spheres this is the lowest achievable reprojection error on the given dataset. However, when applied to the walnut dataset the 1.77 threshold resulted in a rejection rate of 62.3%. For this reason a higher threshold with a lower rejection rate on the walnut dataset is preferred. We have chosen 2.5 pixels as a threshold because when applied to the walnut dataset the rejection rate is very low with 0.8% and the reconstruction results are visually identical to the results of the experiment with the rotational stage as discussed in section 5.4.4.

For the helix sampling rate H (fig. 5.4a) we observed that from 500 to 10.000 the improvement of 111.6 to 88.7 in the mean reprojection error is significant, while the runtime of the calibration per image increases from 0.15 to 4.0 seconds.

Simulating a reduction in the number of spheres on the sample holder (by random sampling without replacement) has the expected consequence of worse reprojection error (see fig. 5.4b). There is no significant difference between 46, 48 and 50 circles

because on most images between 46 and 50 out of 50 circles are detected by the segmentation algorithm.

In fig. 5.4 in the bottom row the calibration algorithm was run with different combinations of both hyperparameters. Blue crosses resemble the calibration result of the helix and red crosses are the detected circles. We can observe that reducing the number of spheres to 10 makes the calibration unusable and increasing it to more than 50 was physically not possible as the circles would start to overlap on the images, especially on the curves of the helix.

5.4.3 Calibration

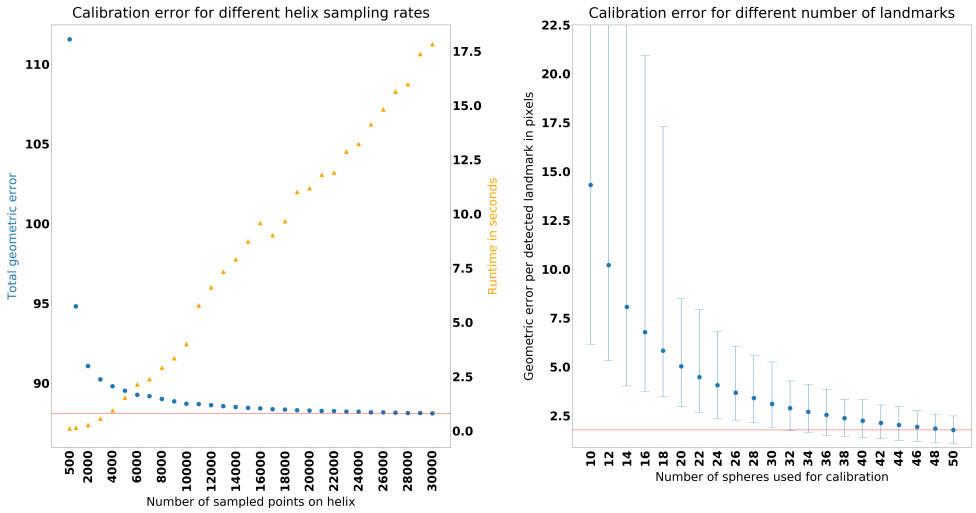
A calibration procedure example is displayed in fig. 5.2. Fig. 5.2 a) is post-processed and the circles are detected. Three circles were not detected. In c) the current estimate for the external parameters were used to project $H = 10.000$ helix points (blue crosses) onto the image. To improve its overlap with the detected circles (red crosses) the parameters are optimized with the above problem statement. Finally the helix is projected onto the image with the improved parameters (see fig. 5.2 d). We can see that the overlap has improved substantially.

5.4.4 CT measurements

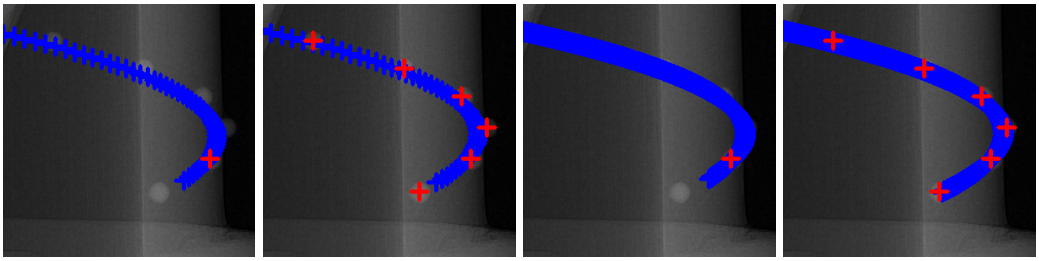
Our observation from fig. 5.5 is that there is no qualitative difference between the results of the robotic sample holder and a conventional rotational stage as a reference. As the volumes could only be registered manually, we perform only a qualitative assessment. Two samples of different size and internal structure were measured, a walnut and a pistachio. We can see in fig. 5.5a that the internal structure of the walnut was mapped as accurately for the reconstruction of the measurement with the robotic arm as with the reference experiment. The pistachio in fig. 5.5b has a simpler structure compared to the walnut, but we can assess the slit in the kernel and the sharpness of the shell. When comparing the reconstruction of the measurement with the robotic arm to the reference reconstruction, we can see that there is no loss in the visual quality of the slit in the pistachio's kernel and its shell in both the top- and front-view.

5.5 Conclusion

In this chapter, we have studied the repeated placement accuracy of the FRANKA EMIKA Panda robotic arm and we have introduced a calibration procedure for solving its accuracy issue. Our intermediate sample holder part houses a helix as calibration structure and it is able to calibrate the position and orientation of the sample sufficiently accurately on the detector images. In the following chapters, we will use the results from this chapter to study the use of the Panda robotic arm for more advanced trajectories (e.g., spherical and runtime optimized trajectories).

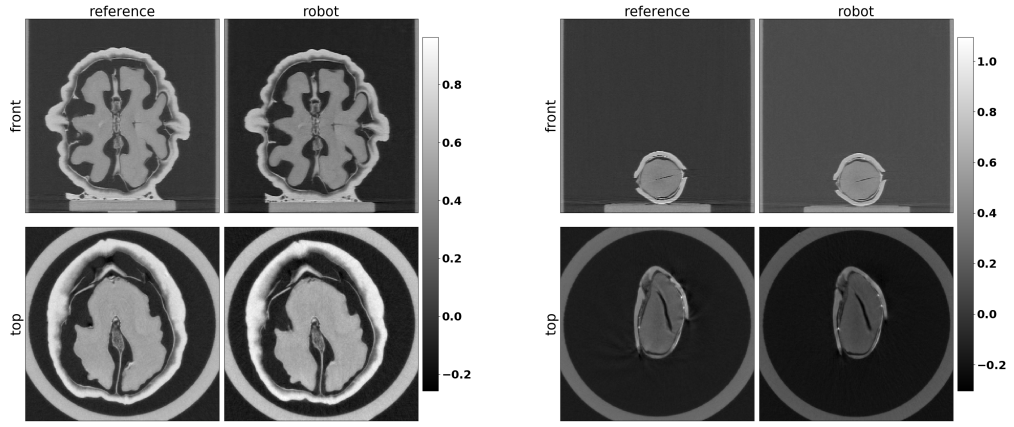


(a) Helix sampling rate (b) Number of spheres



(c) 10 spheres - 500 helix points (d) 50 spheres - 500 helix points (e) 10 spheres - 30000 helix points (f) 50 spheres - 30000 helix points

Fig. 5.4.: Calibration parameters. The calibration is influenced by two parameters: The number of points that are sampled on the helix structure for the distance measurements and the number of spheres on the sample holder. Calibration results were evaluated based on different choices for the two different parameters. The *geometric error* between the sampled helix structure and the detected spheres was used as error metric. A higher sampling rate on the continuous helix structure leads to a lower error but increases the runtime linearly (a). A higher number of spheres decreases the error but physical constraint do not allow to increase this number as there is limited space on the sample holder (b). Calibration results for different combinations of the parameters are displayed for a region of the helix structure (c to f). Decreasing the number of spheres severely affects the calibration results.[Pek+22b]



(a) Walnut experiment (b) Pistachio experiment

Fig. 5.5.: Experimental results. A walnut and a pistachio were measured and reconstructed in order to compare the conventional rotational stage (*reference*) with the robotic sample holder (*robot*). The reconstruction volumes were registered and aligned but small differences are still visible. The detector images were binned with $4 * 4$ and the reconstruction volume has dimensions 720^3. The front slice is from the perspective of the x-ray source. The top slice is from the bird's eye view. A zoom factor of 2x was applied to the slices to crop the region of interest. Our observation is that the reconstruction quality is identical despite the fact that the volumes are not aligned perfectly and hence the contrast does not match.[Pek+22b]

6 Spherical acquisition trajectories for X-ray computed tomography with a robotic sample holder

6.1 Introduction

In this chapter, we study the challenges of executing spherical acquisition trajectories for X-ray CT with the robotic sample holder introduced in chapters 4 and 5. We use the calibration procedure introduced in chapter 5 and adapt the intermediate sample holder part for seamless execution of spherical trajectories in the laboratory environment.

This chapter is an adapted version of a peer-reviewed manuscript [Pek+23b] and a conference publication [Pek+23a] published during my Ph.D.

In our previous publication, we introduced a flexible robotic arm with seven degrees of freedom (DoF) as a sample holder within a laboratory X-ray computed tomography (CT) setup [Pek+22b]. The arm adds flexibility to the setup as a sample holder by enabling arbitrary rotations and placement of the sample. Hence, it allows non-standard trajectories, as opposed to conventional circular or helical trajectories, that are not restricted in their sequence. We also introduced a suitable calibration mechanism to determine the exact positioning of the sample from the image, as the values reported by the sensors of the robotic arm are not precise enough for reconstruction purposes. The calibration mechanism requires a sample holder part attached to the robotic arm, which was also introduced in [Pek+22b].

In the following, we present our work on optimizing various aspects of the robotic sample holder for the seamless execution of spherical trajectories. We modified the sample holder part attached to the robotic arm to improve coverage of spherical trajectories. We provide a detailed analysis of the different shapes of this sample holder part and their effect on the execution of different types of trajectories. We also present experimental results demonstrating the improved performance of spherical

trajectories our system can execute. We provide a quantitative comparison of reconstruction results to conventional trajectories.

In the remainder of this section, we will provide an overview of related work on trajectory optimization methods for imaging systems.

Our sample holder is designed and optimized for executing task-specific trajectories. The authors of [Her+21] use their system to present a trajectory optimization approach that optimizes image quality. Existing work on trajectory optimization relies heavily on simulations of X-ray projections with models of known samples (see [Hat+22]) because of a lack of sample holders that can place the sample at arbitrary rotations. When employing robotic systems that tackle this limitation, the placement accuracy of the robotic arm is not high enough for CT purposes. For the reconstruction of the measurements, we need a suitable calibration mechanism that extracts the geometry of the sample [Pek+22b]. However, the systems mentioned above lack this feature. Our system solves this issue and hence can execute arbitrary trajectory types in practice.

In general, adding non-standard trajectories is advantageous for the measurement results. Using cone-beam geometry for X-ray CT delivers acceptable results but can still arise; for instance, samples with highly absorbing parts will introduce artifacts in the reconstruction. Non-standard trajectories reduce acquisition times or deliver better image quality for an equal number of projections with added information from new acquisition angles. Stayman et al. have shown the advantages of non-standard trajectories extensively in their studies [GSS20b; Rey+23; Ma+22].

In this chapter, we will outline our approach toward executing task-specific trajectories by investigating the challenges of executing non-standard trajectories for spherical trajectories. We will conclude the feasibility of using a robotic arm as a sample holder for arbitrarily complex trajectories and the limitations we must tackle.

6.2 Methods

In this section, we discuss the methods for executing spherical trajectories with the robotic arm as a sample holder in a laboratory X-ray CT setup. After introducing the system's hardware components, we describe more specific aspects like path planning, sphere sampling, and reconstruction. We also refer interested readers to [Pek+22b] for a more detailed introduction to the robotic sample holder for X-ray CT.

6.2.1 Sample holder part

The sample holder part is a critical component of the system as it allows the robotic arm to grasp samples of arbitrary shape and is a fundamental part of the calibration process that identifies the position and orientation of the sample. The sample holder part and the gripper components are visualized in Figures 6.1b, 6.1c, and 6.1d. It consists of two parts. The gripper part is where the robot's fingers can grasp the holder steadily. The cylinder part fulfills the purpose of placing a helix of fiducial markers on a cylinder next to the sample. The lower part is called *gripper part* throughout this paper, and it mounts directly to the last link of the robotic arm after unmounting the hand from the arm. We will discuss reasons for using the arm without the hand in section 6.2.2. Before attaching the sample holder part to the robotic arm, the sample needs to be glued to the mounting plate, which is inserted into the cylinder from the top at the intended position (see Fig. 6.1c).

The cylinder is 118 mm tall and 50 mm in diameter inside. The sample holder part was designed with the 3D modeling software *Autodesk Inventor* and manufactured by a cutting machine from a solid piece of polyoxymethylene (POM).

The reference structure used for calibration is embedded in the sample holder and is a helix comprising 50 embedded aluminum spheres of 2 mm diameter. These spheres were fixed manually in notches included in the holder's design process. The spheres appear as circles on the detector images we segment during calibration.

Compared to our previous work[Pek+22b], we modified the sample holder part by introducing a gripper part that connects the robotic arm's last link with the holder's cylindrical part. This new part enables the use of different gripper shapes (straight, curved) in varying lengths, which enables the robotic arm to reach different areas of more specific trajectories, for example, spherical trajectories. We mount the gripper part to the cylinder with a screwing mechanism.

6.2.2 Path planning and robot control

With the path planning procedure, our system exposes an abstract interface to the user for planning and executing advanced trajectories. A detailed description of the path planning process can be found in our previous publication [Pek+22b]. We have also provided additional implementation details in section 4.2.4.

A trajectory comprises a series of *way-points* that the robotic arm approaches in the given order. The arm stops at each way-point, triggering the detector to capture

an image. After successful image acquisition, the robotic arm continues trajectory execution. The user specifies the parameters for sampling the way-points on the trajectory from the user interface. Given these parameters, the system first samples the way-points depending on the trajectory type. Subsequently, the underlying motion planning pipeline plans a path from each way-point to its successor. If the arm cannot reach way-point $i + 1$ from way-point i, e.g., because there is no collision-free path, we plan a path from i to way-point $i + 2$, and $i + 1$ is marked as not reachable. The unsuccessful planning means that we know that no detector image will be captured later for way-point $i + 1$ before executing the trajectory. Finally, we connect these paths, and the output is a trajectory that starts at the robotic arm's current position and passes all way-points in the given order.

Problems can still arise while executing the successfully planned trajectory. The motion planning pipeline plans the paths between the way-points based on an internal model that the manufacturer provides. This model includes the kinematic and collision model of the robotic arm.

The kinematic model defines the arm's link lengths, joint types, and joint limits. The manipulation software constructs a *configuration space* from this model with n-dimensions, where n is the number of joints. The software samples this space at a fixed density and connects the sampled points with an exploration algorithm. The result is a graph where each vertex represents a collision-free and valid configuration for the robotic arm.

The collision model marks specific areas of the *configuration space* as colliding configurations of the arm, which protects the arm from colliding with itself in specific configurations. These collisions are possible because driving the joints within their valid ranges can easily result in a configuration where certain links collide with each other; for example, the last link can easily reach the first link. The surroundings of the robotic arm are also added to the configuration space (e.g., the detector and the table) as collision objects, invalidating possible configurations of the robotic arm in which it would collide with these objects. We discussed this in our previous work under *passive collision detection* [Pek+22b].

Incorporating these two models into our path-planning process should be enough to plan safe trajectories. In practice, issues arise because the controller box only receives part of the planned trajectory in advance from beginning to end. It sequentially receives joint commands of the current trajectory one by one. As a result, there is uncertainty in the control box about the upcoming commands for the joints, as they might trigger a collision. For this reason, the control box that receives the joint commands one by one interpolates the upcoming command by combining the latest

commands and the current state to check if the arm's links will collide in the next instant. If this check is positive, an internal collision avoidance reflex is triggered, and the control box stops the robotic arm. We trigger the error recovery routine provided by the manufacturer in order to recover from this state. The links are moved slightly out of our planned trajectory for this purpose. We have increased the sizes of the robotic arm's links in the internal collision model of the robotic arm in order to reduce the number of these reflexes.

The error recovery routine mentioned above leads to an undefined state in our planned trajectory because the resulting position of the arm is unexpected, as it lies outside of the planned trajectory. We have implemented an error recovery routine for our trajectory that will plan to a fixed intermediate way-point (*homing state*) and set a flag to ensure we do not use the image recorded there in later steps of our pipeline. Afterward, we plan from the homing state to the next scheduled way-point, where the handover to the initially planned trajectory will occur.

Our software package calculates a unique identifier for the current state of the robotic arm's environment, the *configuration space*. It also calculates a unique identifier for the trajectory that the user requested. Both identifiers are calculated with a *sha256* hash of the requested parameters (trajectory) and contained elements (configuration space). This calculation is an additional safety measure as it maps different compositions of the robotic arm's environment to a unique identifier. We can save a planned trajectory to the file system for repeated use in the same environment with a unique ID for the trajectory. When the user requests a trajectory with the same parameters as before, we check for the file on disk and send the trajectory directly to the robotic arm for execution. Additionally, the trajectory ID ensures that in combination with the environment ID, we only execute a trajectory from the filesystem when it suits the current environment, including the surroundings of the robotic arm, the sample holder, and the attached sample.

6.2.3 Sphere sampling

In order to generate a spherical trajectory, we need to sample points covering the surface of a sphere. Each point represents a rotation of the sample.

Sampling a fixed number of n points on the sphere is a well-studied problem [SK97; Raf19; KKM14; HS95]. The goal is to distribute the points uniformly on the sphere's surface. Trivial approaches like sampling on the two polar axes independently and combining the samples to get 3D coordinates do not lead to a uniform sampling on

the sphere surface. In this work, we utilized the Hierarchical Equal Area isoLatitude Pixelization (*HEALPix*) for this purpose [Gor+05].

HEALPix partitions the sphere's surface into a fixed number of areas of equal size. The centers of these areas are the sampled points on the sphere. The discretization number N_{pix} determines the number of points on the sphere. In HEALPix, the grid resolution parameter N_{side} determines the number of pixels and hence the number of points sampled on the sphere: $N_{pix} = 12 \cdot N_{side}^2$, where $N_{side} \in \mathbb{N}$. The user chooses N_{side}, and N_{pix} is calculated from that parameter.

If the user needs a specific number of points on the sphere, we can choose a higher grid resolution parameter N_{side} in the first step, and a smaller number of points can be sampled from the resulting grid in a later step. For our experiments, this did not present a problem as we chose $N_{side} = 10$, which results in $N_{pix} = 12 \cdot 10^2 = 1200$ pixels on the grid and hence 1200 potential way-points on the spherical trajectory.

6.2.4 Calibration

The calibration procedure tackles the issue that the robotic arm does not sufficiently accurately place the sample at the desired position due to inaccurate path planning and inaccurate electrical motors at its joints. Reading the sensors of the robotic arm and deducing the sample's current position is also insufficient to determine the correct position, as the sensors report inaccurate values. However, the reconstruction step requires the exact position of the sample at each view. With the calibration procedure, we can identify the actual positions and orientations of the sample from the detector images. For the calibration, a sample holder part with an embedded geometric reference structure we can identify on the detector images was necessary and introduced in section 6.2.1.

The sample holder houses a helix structure with known parameters. A set of points was sampled on this continuous structure to place calibration spheres at known locations (drilled holes on the cylinder part), which we will later segment on the detector images. We determine the expected location of the helix structure on the detector images with a priori information about the system: the camera matrix, the robotic arm's location in the setup, and the values reported by the sensors at the arm's joints. We can project the continuous helix structure onto the image and use a suitable cost function to determine the error between our guess for the helix structure and its actual position on the image by comparing it to the segmented circles. We optimize the parameters for the translation and rotation of the geometric

structure and the sample on a given image with the Levenberg-Marquardt non-linear optimization algorithm. The sensor values from the robotic arm serve as an initial guess for the optimization. A more detailed description of the calibration procedure is available in [Pek+22b].

We have improved the calibration procedure by checking the segmented circles for false positives. While testing the system with different samples in our lab, we experienced that the segmentation algorithm detected circles falsely on the screw heads and the samples. We solved this issue by checking the distance (L2-Norm on 2D image) between the segmented circle centers and our guess for each calibration circle. Suppose the distance of the given segmented circle's center is below a certain threshold for one of the guessed calibration circles. In that case, we interrupt the checks and assume that the segmented circle is a calibration circle. Suppose the calculated distance is above the fixed threshold for all calibration circles. In that case, we assume that this specific circle is a false positive, which we do not consider in the later steps of the calibration procedure. For example, for the experiment with the curved sample holder part on the spherical trajectory, identifying the false positives improved the number of successfully calibrated images from 912 to 946 out of 965.

6.2.5 Reconstruction

We used around 900 equidistant X-ray projections for the tomographic reconstruction along a circular or spherical trajectory sized 720×720 pixels with a spacing of $600\,\mu m$. For the spherical trajectory, the number of projections varied by $\pm 8\%$ (see Table 6.1a). The remaining parameters of the reconstruction were as mentioned in section 3.3.

6.3 Experiments and results

When running the main experiments, we designed two different gripper parts for the sample holder part and analyzed their coverage (*reachability*) for different trajectory types with the robotic arm. Additionally, we modified the mounting mechanism of the robotic arm by removing the hand and gripper from the arm and mounting the sample holder part directly to the last link. The cylindrical sample holder part from section 6.2.1 was used for all experiments for geometric calibration. The collision detection algorithm was running in the background throughout these experiments.

6.3.1 Reachability analysis

The term *reachability* describes the percentage of target way-points that the robotic arm can reach from all target way-points. For example, for a spherical trajectory with a fixed number of uniformly sampled points on the sphere, the reachability states the percentage of points that the robotic arm can *reach*, meaning that the motion planning pipeline could find a valid and collision-free configuration for the arm for that specific point and a path to that configuration.

We have conducted experiments with two types of trajectories: circular and spherical. We have used two different grippers for each trajectory type, resulting in four experiments with an identical sample. Our goal with these experiments was to analyze the impact of the gripper types on the reachability of the two trajectory types.

6.3.1.1. Sample holder gripper type

We manufactured and used two different types of gripper parts for the sample holder part mentioned in section 6.2.1: straight and curved (l-shaped, $90°$). Figures 6.1b and 6.1d show the grippers. The straight gripper differs from the sample holder part used in our previous publication because it introduces an additional distance between the grippers and the cylindrical part of the sample holder. When we neglect the gripping cave, the length of the straight part amounts to 70 mm. We expected improved reachability of spherical trajectories by introducing extra distance between the robotic arm's last link and the sample mounted on top of the cylindrical part and positioned at the central X-ray. The extra distance would increase the distance between the individual links of the arm when approaching a goal for specific configurations, which helps avoid collision reflexes of the arm.

6.3.1.2. Trajectory type

We executed our experiments on circular and spherical trajectories. We aimed to compare the proposed robotic system's performance on these trajectory types for reachability. For the spherical trajectories, the sphere discretization parameter N_{side} was set to 10, which resulted in 1200 points on the sphere. For the circular trajectories, we sampled 900 points on the circle in order to match the number of reachable points on the spherical trajectory for better comparability of our results, as the robotic arm can reach almost all way-points on the circular trajectory compared

Tab. 6.1.: Trajectory poses statistics

	circular	spherical
straight	900 / 900 (100 %)	909 / 1200 (75.6 %)
curved	835 / 900 (92.8 %)	979 / 1200 (81.6 %)

(a) **Reachability statistics**: Way-points with successful motion planning / potential way-points.

	circular	spherical
straight	900 / 900 (100.0 %)	867 / 909 (95.4 %)
curved	835 / 835 (100.0 %)	965 / 979 (98.6 %)

(b) **Trajectory execution statistics**: Successful execution by robotic arm / way-points with successful motion planning.

	circular	spherical
straight	896 / 900 (99.6 %)	860 / 867 (99.2 %)
curved	833 / 835 (99.8 %)	946 / 965 (98.0 %)

(c) **Calibration statistics**. Successful calibration / successful execution by the robotic arm.

to the spherical trajectory where the robotic arm cannot reach approximately a quarter of the way-points.

We have created detailed statistics on the failure rates of all experiments during each stage of our pipeline. The most common issue is that the motion planning pipeline can not find a valid configuration for a way-point on the trajectory. Other issues that can reduce the number of usable images for the reconstruction are collision reflexes triggered by the robotic arm (section 6.2.2) and erroneous calibration (section 6.2.4).

We have created tables with detailed statistics on the three types of errors. The number of reached way-points compared to the potential number of way-points are listed in table 6.1a. The number of way-points the arm could reach without triggering reflexes during execution is listed in table 6.1b. The number of acquired images for each successfully calibrated experiment is listed in table 6.1c. For example, for the measurement with the straight gripper part and spherical trajectory, the system acquired and calibrated 860 images out of 1200 potential way-points.

In Fig. 6.2, we plotted each experiment's reachability as a two-dimensional coverage map. The trajectory way-points represent rotations of the sample as 3D points on the sphere surface. We projected these points to 2D with the cartographic Mollweide projection [**snyder1997flattening**] for improved visualization of the coverage of the surface area.

We differentiate the two trajectory types, circular and spherical. The maps visualize coverage maps on the rows for the two gripper types (straight and curved). While the interior of the projected sphere surface includes all possible rotations in 3D and hence all rotations of the sample, only those rotations where the robotic arm has a valid and collision-free configuration are plotted with cross markers. The empty (white) spots resemble rotations the arm cannot reach. It is important to note that the circular trajectory only attempted a fraction of the rotations on the sphere surface because of the trajectory type.

We have also plotted two additional reachability maps where we simulated two scenarios only possible in the simulation (see Fig. 6.3). For the first plot, we removed all obstacles from the configuration space except for the table, which should represent the ideal situation where a hutch is constructed specifically for the robotic sample holder. For the second plot, we also removed the X-ray beam collision object from the configuration space to calculate the highest possible coverage for the robotic arm. The X-ray beam is modeled as a collision because it avoids occlusions of the arm with the sample on detector images by invalidating such configurations of the robotic arm.

6.3.2 CT measurements

We conducted four experiments: one sample was measured with two different gripper parts (straight and curved) for the sample holder on two trajectories (circular and spherical).

The sample we used for all experiments is displayed from two perspectives in Fig. 6.4. It consists of two separate parts: a bunny toy brick (Fig. 6.4b, left) and a solid piece of polyvinyl chloride (PVC) with a thickness of 4 mm (Fig. 6.4b, right). We chose this composition because these two parts differ significantly in their absorption rate, which helps compare circular and spherical trajectories in their reconstruction performance for image quality. We have cut the absorber plate in a non-orthogonal shape relative to the mounting plate and arranged it next to the toy brick to cause beam-hardening artifacts in the reconstructions of our experiments with this sample. One can execute the circular trajectory with conventional methods like a rotational stage, but a flexible sample holder like ours is necessary to execute the spherical trajectory. We aim to demonstrate the superiority of spherical trajectories for complex samples.

For each CT measurement, we acquired the images with a source voltage of 45 kV, source power of 1445μA, and exposure time of 1s. In Fig. 6.5a, the reconstruction of

our sample is shown from three different perspectives (YX, YZ, and ZX) for the two trajectory types and the straight gripper part. All volumes are registered with each other due to the calibration process[Pek+22b], as the center of the helix structure serves as the coordinate system's origin. We plotted line profiles in Fig. 6.5a for three different cross-sections of the central YX slice of the sample.

6.4 Discussion

In this section, we will discuss the results of our experiments from section 6.3, where we aimed to measure our system's performance for trajectory reachability and reconstruction image quality. We categorized the experiments by trajectory and gripper part type. The trajectory type affects the reconstruction image quality and completeness. The choice of gripper part type affects the system's ability to reach specific goals and the probability of triggering collision reflexes of the robotic arm's control box.

6.4.1 Reachability analysis

In Fig. 6.2, we plotted each experiment's reachability as a two-dimensional coverage map. Additionally, we have listed the absolute and relative reachability and trajectory execution success numbers in tables 6.1a and 6.1b.

6.4.1.1. Sample holder gripper type

Choosing different types of grippers results in different unreachable regions of the sphere. For both gripper types, we have an explanation for the two blind spots near the equator: We have modeled the X-ray beam as a cubic collision object in the configuration space of the robotic arm in our motion planning pipeline. This modeling means that unless contact of a specific link with the X-ray beam is explicitly allowed as an exception, all configurations of the arm that place one of the links into the X-ray beam are marked as a potential collision and hence as invalid configurations. At these rotations, one of the links would cover the sample on the detector image if an image was acquired.

Two additional issues arise more often with the straight sample holder gripper. The first one is that collision reflexes are triggered more often when approaching spots

on the bottom of the sphere because the last link of the arm has to approach the way-point at a spot close to the first link. With the curved gripper, we can increase the distance between the first and last link and limit this behavior. The second issue is that way-points that lie at the opposite end of the arm's mounting position are harder to reach for the arm with increasing sample size as the maximum reach of the arm is physically limited by the link lengths. The dependence on the sample size occurs since we center the sample at the central X-ray, which puts the last link of the arm further away from the central X-ray.

6.4.1.2. Trajectory type

We can see that the robotic arm has no difficulties reaching every single point on the circle with the straight gripper part for the circular trajectory. The circle is parallel to the table where the arm is also mounted. With the curved gripper, the arm cannot reach a fraction of the circular trajectory (7.2%, see table 6.1a) as the links collide with the X-ray beam collision object (see section 6.4.1.1) which would result in occlusions on the detector images. In contrast, we can see that the robotic arm cannot reach way-points in some areas of the sphere for the spherical trajectory, meaning that the reconstruction will lack images from specific rotations of the sample. The unreachable region's location and size depend on the gripper's choice. The potential coverage rate lies between 75.6% and 81.6% out of 1200 potential way-points.

Furthermore, we can conclude from the numbers in tables 6.1a and 6.1b that with the right choice of sample holder gripper type, the actual coverage of the sphere surface can be further increased when considering issues that arise during trajectory execution (see section 6.2.2). The coverage rates of 75.6% and 81.6% mentioned above decrease further while the robotic arm executes the planned trajectory: The arm only reaches 95.4% and 98.6% of the given coverage rates due to collision reflexes of the robotic arm explained in section 6.2.2. However, we can see that the gripper type affects the error rate during execution significantly, as with the curved gripper, the robotic arm only misses out 1.4% of the trajectory.

In Fig. 6.3, we have plotted theoretical reachability maps that are only possible in simulation. We can conclude from Fig. 6.3a that the reachability could be increased from 81.6% to 84.2% by installing the robotic arm in an extensive safety hutch with more space inside. Furthermore, Fig. 6.3b shows that the given robotic arm can reach 99.1% of the sphere surface with the curved gripper type if we tolerate occlusions of the sample by the links on the detector images. In our experiments,

we do not, because occlusions with the cylindrical part of the sample holder would prevent successful calibration and occlusions with the sample would cause artifacts in the reconstruction.

6.4.2 CT measurements

From a qualitative and quantitative perspective, we can discuss our system's performance by examining the results in Figure 6.5.

Qualitatively, we can see in Fig. 6.5a that the slices depicted in the top row (spherical trajectory, straight gripper) are sharper overall when compared to the slices in the bottom row (circular trajectory, straight gripper). When examining the region between the absorber at the top and the toy brick in the middle (YX-slices), we can see that the slice of the experiment with the spherical trajectory does not cause artifacts. Hence this area is genuinely black compared to the slice on the bottom, where we can spot white traces. We can also spot significant differences for the slices in the center and right columns of Fig. 6.5a, for example, on the right column, the inner structure of the toy brick is much sharper for the spherical trajectory (top right) when compared to the circular trajectory (lower right).

For a more quantitative comparison of the reconstructions, we have plotted line profiles at three different locations of the YX-slices in Fig. 6.5b. The first line profile crosses the absorber, the toy brick, and one of the screws used for mounting the sample plate to the cylinder. The second line crosses the absorber and the toy brick, and the third line only crosses the toy brick.

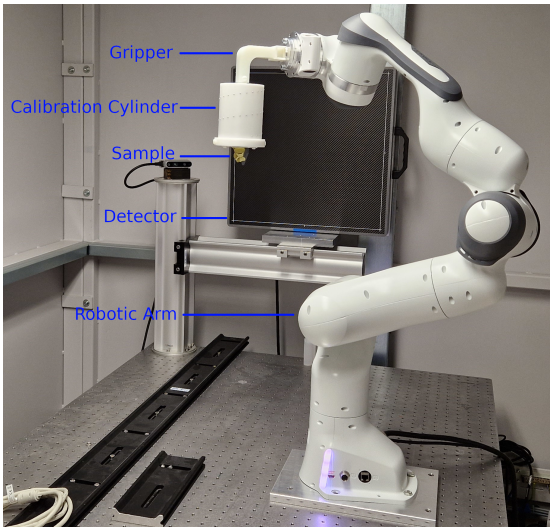
For line profiles one and two, we can see that the red line (spherical trajectory) has a steeper curve and hence a higher derivative for the absorber compared to the blue line (circular trajectory), which means that the image is sharper with the spherical trajectory. Additionally, we can see that at the end of the first line profile plot, the red line can reflect the screw head in contrast to the blue line, where the screw head does not have a notch. We verified the visual increase in sharpness by measuring the gradient magnitude of the line profiles in Fig. 6.5b. The gradient magnitudes of the line profiles 1,2 and 3 improved by 50%, 58% and 80% respectively with the spherical reconstruction compared to the circular.

Another critical observation is that the absorber causes artifacts with two different trajectory types in two orthogonal directions. In the case of the circular trajectory, the artifacts are parallel to X-ray beams. The artifacts are parallel to the absorber and orthogonal to the X-ray beams for the spherical trajectory.

Furthermore, in table 6.1c, we can see that the calibration success numbers are between 98.0% and 99.8% for all experiments. Achieving a high success rate in the calibration is crucial, as only those images with a successful calibration are usable in the reconstruction step. Hence, the calibration influences the reconstruction image quality and completeness. In this case, we can see that the calibration success rates are close to 100%, and in absolute numbers, we missed the highest number of images in the case of the spherical trajectory and the curved gripper part with 19 out of 965 images. 10 out of 19 images capture poses that place the cylindrical part of the sample holder part containing the helix structure parallel to the X-ray beam, where calibration becomes very difficult because the spheres overlap with each other on the image and cannot be segmented. The curved gripper makes these poses easier to reach as they would otherwise cause occlusions by placing the arm's links into the picture. Our motion planning pipeline avoids these poses, especially for the straight gripper, by modeling the X-ray beam as a cubic collision object (see section 6.2.2). Hence the inferior calibration statistics with the curved gripper.

6.5 Conclusion

In this chapter, we have studied the execution of spherical trajectories with the robotic sample holder extensively. To achieve this goal, we have adapted the intermediate sample holder part and different parts of our software package (e.g., motion planning and trajectory serialization). In the next chapter, we will use the ability of our robotic sample holder to sparsely sample the spherical trajectory at runtime based on the absorption characteristics of the given sample without prior knowledge. The sparse sampling will enable runtime optimized acquisition trajectories and improved reconstruction image quality.



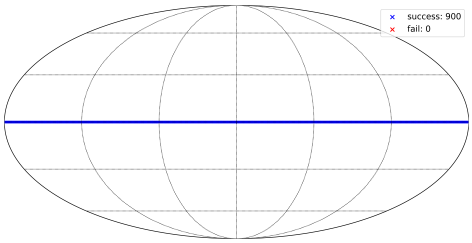
(a) Lab setup

(b) Straight gripper

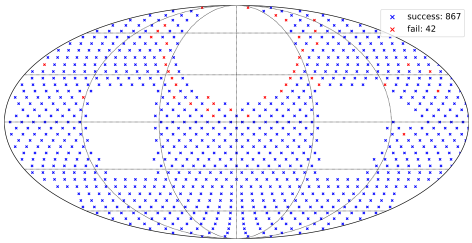
(c) Cylinder part with calibration structure (helix)

(d) Curved gripper

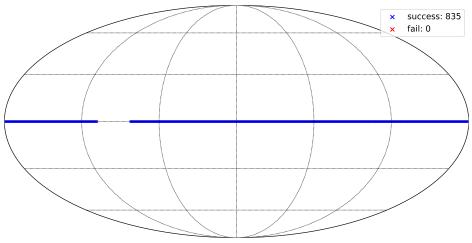
Fig. 6.1.: Hardware setup. In (a), the robotic arm is mounted on a table with the source and the detector inside a safety hutch. The source-to-robot distance is 136 cm, and the robot-to-detector distance is 79 cm. Two depth cameras monitor the robot's movement and stop the robot controller when the executed trajectory interferes with obstacles. A power switch can also stop the robotic arm. It routes to the operator's table outside of the hutch. In (b), (c), and (d), the different parts of the sample holder part are displayed. (b) and (d) depict the two gripper types, and (c) depicts the cylinder part, which houses the geometric structure for calibration. The gripper part is mounted to the cylinder part for experiments.[Pek+23b; Pek+23a]



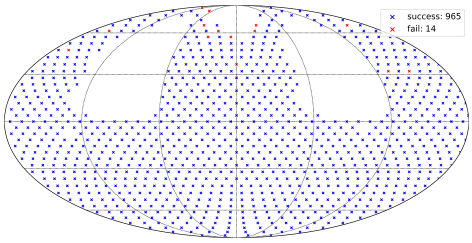
(a) Straight gripper - circular trajectory



(b) Straight gripper - spherical trajectory

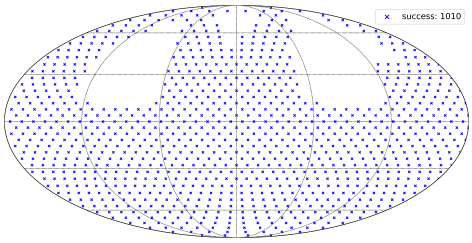


(c) Curved gripper - circular trajectory

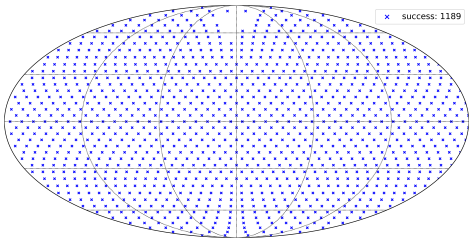


(d) Curved gripper - spherical trajectory

Fig. 6.2.: **Reachability**. Reachability maps are plotted for each trajectory type (circular vs. spherical, columns) and each gripper type (straight vs. curved, rows). The plots resemble a 2D cartographic projection (Mollweide projection) of the 3D sphere surface. The circular trajectories cover a fraction of the sphere surface, representing all possible sample rotations. The robotic arm can reach all points on the circle (left column). For the spherical trajectory, the robotic arm cannot reach all way-points, hence the blind spots on the maps (right column). The red points indicate way-points with successful motion planning but failure during execution (see section 6.2.2 for details). We achieved the best coverage of the sphere (81.6 %) with a curved gripper for the sample holder (lower right).[Pek+23b]

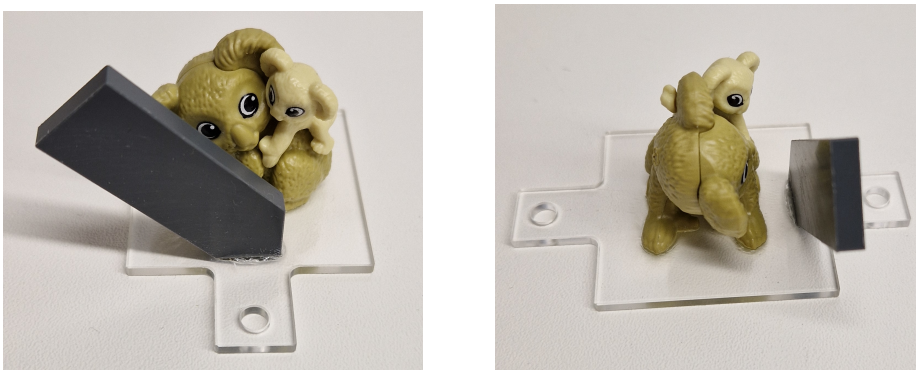


(a) Spherical trajectory, curved gripper - no obstacles



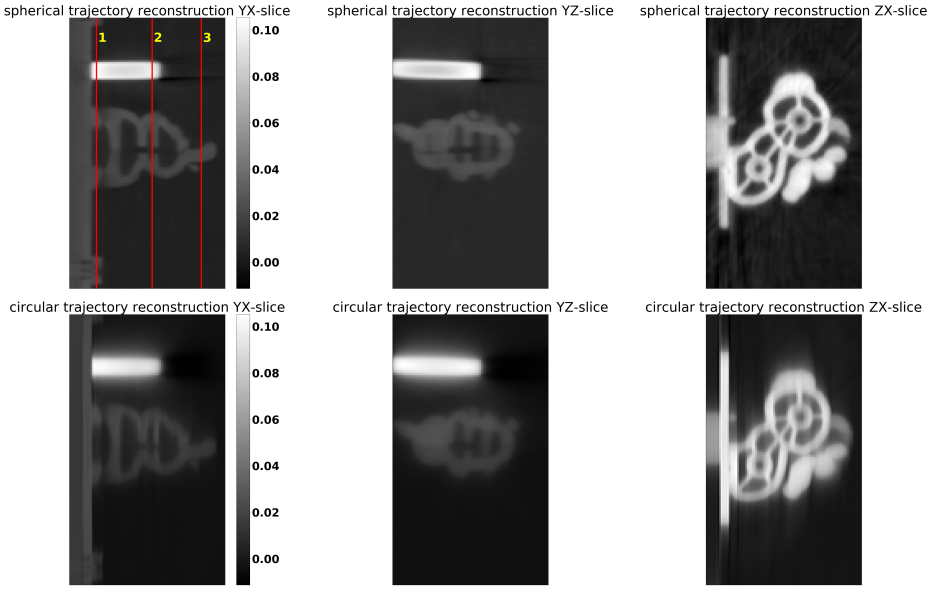
(b) Spherical trajectory, curved gripper - occlusions of sample by arm allowed

Fig. 6.3.: **Theoretical Reachability**. We plotted theoretical reachability maps for spherical trajectories for two situations only possible in the simulation. On the left, we removed all obstacles surrounding the robotic arm in the laboratory setup from the configuration space except for the table. On the right, in addition to removing all collision objects, we removed the x-ray beam encoded as a cubic collision object between the source and the detector. This collision object prevents occlusions of the sample by the robotic arm on the detector images. On the left, we achieved a successful planning rate of 84.2% (1010/1200), and on the right, 99.1% (1189/1200). The reachability number 99.1% proves the flexibility of the given robotic arm in combination with the curved gripper.[Pek+23b]

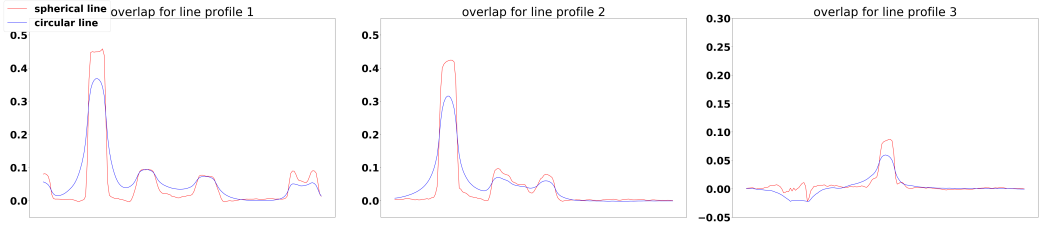


(a) Sample side (b) Sample front

Fig. 6.4.: Sample. The sample consists of the object of interest (a toy brick) and an absorber (polyvinyl chloride plate), which were manually glued to a Plexiglass mounting plate. The toy brick has dimensions 31 x 21 x 31 mm, and the absorber has a thickness of 4 mm. The absorber plate has a significantly higher X-ray contrast absorption rate than the toy brick. We aim to introduce beam hardening artifacts with this property in the reconstructions and evaluate the performance of different trajectory types in tackling this issue.[Pek+23b; Pek+23a]



(a) Reconstruction slices



(b) Reconstruction line profiles

Fig. 6.5.: Experimental Results. A sample was measured and reconstructed with the robotic arm with the straight gripper part to compare the circular trajectory (conventional) with the spherical trajectory (advanced). The reconstruction volumes are registered and aligned with our calibration algorithm (see section 6.2.4). We binned the detector images with $4 * 4$, and the reconstruction volume has dimensions 720^3. A zoom factor of 5x was applied to the slices to crop the region of interest. We plotted line profiles at three different positions for the YX slices. Our observation is that the reconstruction of the measurements with the spherical trajectory (top left, red lines) is superior qualitatively and quantitatively compared to the reconstruction of the circular trajectory (bottom left, blue lines). Qualitatively, there are fewer artifacts, and the image is sharper than with the circular trajectory. From a quantitative perspective, the line profiles for the spherical trajectory (red) are steeper than those for the circular trajectory (blue), making the image sharper.[Pek+23b]

7 Runtime optimization of acquisition trajectories for X-ray computed tomography with a robotic sample holder

7.1 Introduction

This chapter studies the runtime optimization of spherical acquisition trajectories for X-ray CT with the robotic sample holder. We use the methods and algorithms that we developed in chapters 5 and 6 to develop a new method for sparsely sampling the "ideal" points on the sphere for the given sample. In the following sections, we will outline our method for measuring samples without prior knowledge about their material composition and interior structure.

This chapter is an adapted version of a peer-reviewed manuscript [Pek+23c] published during my Ph.D.

Tomographic imaging systems are expected to work with a wide range of samples that house complex structures and challenging material compositions, which can influence image quality in a bad way. Complex samples increase total measurement duration and may introduce beam-hardening artifacts that lead to poor reconstruction image quality. This work presents an online trajectory optimization method for an X-ray computed tomography system with a robotic sample holder. The proposed method reduces measurement time and increases reconstruction image quality by generating an optimized spherical trajectory for the given sample without prior knowledge. The trajectory is generated successively at runtime based on intermediate sample measurements. We present experimental results with the robotic sample holder where two sample measurements using an optimized spherical trajectory achieve improved reconstruction quality compared to a conventional spherical trajectory. Our results demonstrate the ability of our system to increase reconstruction image quality and avoid artifacts at runtime when no prior information about the sample is provided.

In the following sections, we use the results from our previous publications (see [Pek+22b; Pek+23b]) to improve reconstruction image quality further when using the seven DoF robotic arm as a sample holder by introducing optimized spherical trajectories. We aim to introduce a method that is easy to use and does not require prior knowledge about the provided sample. In addition, we aim to provide an implementation that runs in real-time without affecting the scanning times achieved when acquiring spherical trajectories with the same robotic sample holder. In the following, we present our work on trajectory optimization at runtime for unknown samples with the robotic sample holder. We provide experimental results with two different samples demonstrating the ability of our system to improve reconstruction image quality. We also provide a quantitative comparison of reconstruction results to conventional spherical trajectories.

In the remainder of this section, we will provide an overview of related work on trajectory optimization methods for imaging systems with and without robotic arms.

Optimized non-circular orbits with no prior knowledge of the 3D geometry of the sample were introduced in [Wu+20] for metal artifact avoidance, where the authors presented a method to obtain optimized orbit trajectories for setups with C-arms. Their method was based on a coarse back projection obtained from low-dose scout views and its subsequent segmentation using a U-Net that was trained on simulated data. After this segmentation, the X-ray spectral shift for all possible views in the setup was predicted. The orbit in which this spectral shift was minimized was identified, ensuring the reduction of beam hardening artifacts caused by metal parts. Our work presents optimized non-circular orbits for a more complex setup with a seven DoF robotic arm rather than a C-arm. Moreover, the method we present optimizes the robot's trajectory during runtime, using a scout scan as initialization but using all available information after each iteration to improve the optimized trajectory. Furthermore, our optimization is not based on the X-ray spectral shift, meaning that no information about the absorption coefficient of the material is required.

Another approach for non-circular orbits was presented in [GSS20a]. In this case, the authors used Tuy's condition for data completeness to design orbits that reduced metal artifacts. An approximate location of the metal in the sample was needed to design these orbits. The aspect in which our method differs most is the ability to create the trajectory at runtime, leading to an optimized trajectory that is more adjusted to the actual location of the metal parts of the sample.

In the remainder of this chapter we will outline our approach for executing sample-specific trajectories without requiring prior knowledge about the sample. We will demonstrate the ability of our algorithm to avoid unfavorable acquisition angles when using a robotic arm as a sample holder for arbitrarily complex trajectories.

7.2 Methods

This section discusses the methods for generating optimized trajectories with the robotic arm as a sample holder. Our system generates the trajectories at runtime without apriori knowledge about the provided sample. The algorithms implemented herein can be applied to the laboratory environment or in simulation. After introducing the simulation environment and an overview of the optimization pipeline, we describe more specific aspects like reconstruction volume segmentation, score update, and trajectory pose sampling. We also refer interested readers to our previous publications [Pek+22b; Pek+23b] for a more detailed introduction to the robotic sample holder for X-ray CT.

7.2.1 Pipeline overview

Figure 7.1 depicts our trajectory optimization procedure. The optimization algorithm is split into three parts executed in succession.

In the first step (**I**), the *scout scan* executes a short spherical trajectory and acquires and calibrates images (2 and 3) of the reachable poses. This trajectory is generated based on a sphere sampling pattern with low density (1) to capture the sample from all possible rotations with a minimal set of acquisitions.

In the second step of our algorithm (**II**), the calibrated images from step one are reconstructed (4) with a coarse resolution to obtain an approximate representation of the sample. We segment (5) this representation for determining highly absorbing parts of the sample. Lastly, we calculate a score (6) for each successful acquisition angle a_1 to a_n.

In the last step of our algorithm (**III**), the pool of successfully acquired images is initialized with the images from the results of the scout scan (**I**). The sphere data structure is initialized with the scores from the results of the score calculations (**II**). In contrast to the previous steps (**I** and **II**), the sphere data structure is initialized with a higher density. The increased sphere density samples a higher number of

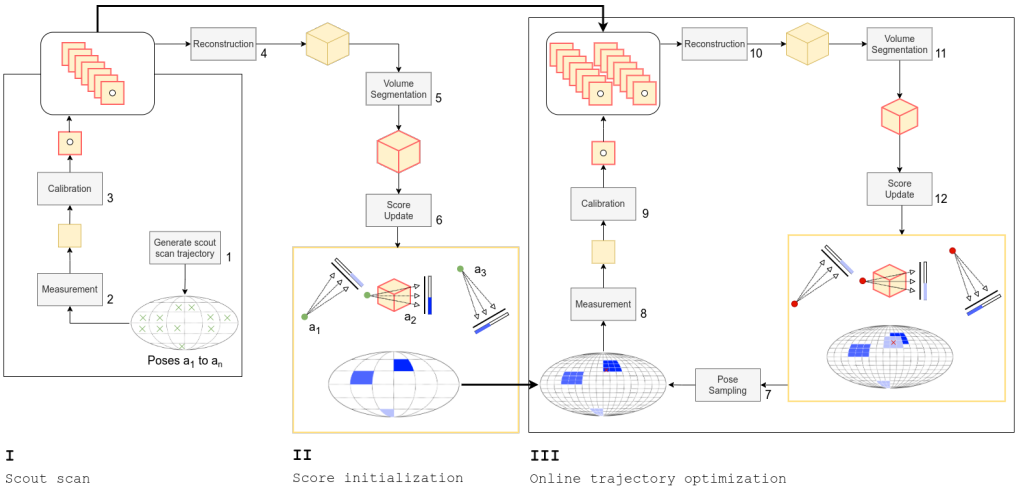


Fig. 7.1.: Online trajectory optimization pipeline. In **I** (1 - 3), the scout scan procedure starts by generating a spherical trajectory $(a_1 \dots a_n)$ that covers the whole sphere with a coarse sampling, only including poses that are reachable by the robotic arm. Measurements of the sample are either measured experimentally or simulated with our robotic software package, and the resulting sinograms are calibrated, extracting the exact geometry of the sample on the image. In **II** (4 - 6), the pool of images acquired in **I** is used to reconstruct the sample in a volume with coarse resolution and to segment highly absorbing regions subsequently. Step **II** is completed by computing a score for each of the poses a_i on the coarsely sampled sphere from step **I**. In step **III** (7 - 12), we enter the open-ended trajectory optimization loop where a densely sampled sphere surface is initialized with the scores from **II**. Likewise, the acquired image pool is initialized with the measurements from **I**. In every iteration of the optimization loop, a new pose is sampled on the sphere surface for measurement, calibration, and volume reconstruction and segmentation by our software package. In the final step (12), a score update is initiated on the sphere for the most recently acquired pose and its neighbors.

poses for more image acquisitions from similar sample angles. The optimization loop starts by sampling a pose on the sphere and acquiring a sample measurement either experimentally or in simulation based on the current coarse reconstruction. The resulting image is calibrated and added to the existing pool of images. The images are used to reconstruct the sample with a coarse resolution (10) and segment highly absorbing parts in the next step (11), like in the score initialization step (**II**). Finally, we recalculate the scores based on the new volume segmentation for all successfully attempted poses up to this point and update the spherical data structure with the new scores (12). The optimization loop continues by sampling a new pose for measurement from the updated spherical map.

The upcoming sections provide detailed descriptions of the individual steps of the online trajectory optimization pipeline.

7.2.2 Scout scan

With the scout scan procedure, we identify the sample's highly absorbing regions on the spherical trajectory before the optimization loop is entered. By coarsely sampling the sphere surface and generating a short spherical trajectory, we acquire images from all possible angles of the sample in a concise amount of time. The main advantage of the scout scan procedure is that highly absorbing regions of the sample are detected with a minimal number of acquisitions. Information about these areas is used in later steps of the optimization pipeline to ensure that the available measurement time is used for less absorbing regions of the sphere. The acquired images are fed into the trajectory optimization pipeline and serve as an initial set of images for the optimization process.

A vital parameter of the scout scan procedure is the number of poses sampled on the sphere. The number of poses determines the scout scan's trajectory size and depends on two factors. First, the sphere discretization number determines the number of equally sized areas into which the sphere surface is cut. The centers of these areas parametrize the rotations of the target poses, and their sum is the size of the spherical trajectory. Second, the reachability of the given robotic arm determines which of the sampled poses can be reached by the given robotic arm. Reachability is a well-defined but complex metric dependent on a range of factors, such as the physical properties of the arm and its operating environment. We extensively study the reachability of the seven DoF robotic arms for a wide range of operating conditions in our previous work [Pek+23b].

We experimented with different values (12, 48, and 108) and decided to use 108 for our experiments. The number cannot be chosen freely as it depends on the grid resolution parameter N_{side} that is required by the HEALPix library, which we use for spherical partitioning. Given the reachability score of our robotic arm of 82% [Pek+23b], the arm would reach 10, 40, and 89 poses. Higher numbers could also be considered, but they significantly increase the allocated time for the scout scan procedure; for example, the higher resolution of $N_{side} = 4$ results in $192 * 0.82 = 157$ measurements until the optimization loop is entered.

The spherical trajectory of the scout scan is planned by attempting each of the sampled poses individually with the motion planning pipeline. If the pipeline cannot find a valid and collision-free path to a sampled pose, it is discarded and marked unreachable in the spherical data structure. Once a valid path is found for the next pose, the robotic arm executes this part of the trajectory. Once the arm reaches the pose, it stops to acquire an image before attempting to plan a path to the next

pose on the sphere. The acquired images are segmented and calibrated with the methods introduced in [Pek+22b] to extract the exact geometry of the sample for the reconstruction step. The calibrated image is added to the pool of successfully acquired images.

7.2.3 Score initialization

In the score initialization step, we perform the operations necessary to transfer the results of the scout scan procedure to the optimization loop.

We reconstruct the sample inside a volume with a coarse resolution with $288 \times 288 \times 288$, which corresponds to a tenth of the detector's resolution. Choosing a small resolution for the reconstruction allows us to perform the following steps in the pipeline on demand. The volume segmentation and the score update can be executed in real time when the reconstruction volume has the abovementioned resolution.

The volume segmentation is applied to the resulting reconstruction volume, which contains the sample holder and the sample. The segmentation output is a volume where the sample holder is filtered out, and only the highly absorbing parts of the sample are visible. The segmentation is described in detail in section 7.2.4.

The score update step is performed on the output of the volume segmentation for all poses individually. An absorption score is calculated for each pose, and the resulting score is saved in the spherical data structure for the part of the sphere surface it represents. The spherical data structure used for this purpose has a low sampling density. Later, the score will determine the likelihood of the pose's neighborhood to be sampled as the next pose. The score update step will be discussed in detail in section 7.2.5.

In the final step, the scores from the sphere with a low sampling density are transferred to a new sphere with a higher sampling density. We need the new sphere for sampling more poses from similar angles. For the score transfer, the score of a specific pose on the sphere with low sampling density is set for all poses it covers on the sphere with higher sampling.

The image pool of the optimization loop is initialized with the measurements from the scout scan procedure. These can provide an initial set of images for the image reconstructions in the first few iterations of the optimization loop. The reconstructions in the first iterations would have severe artifacts if the image transfer did not occur.

7.2.4 Volume segmentation

In order to optimize the acquisition trajectory based on the absorption characteristics of the sample, the projections containing a significant area of the high-absorbing regions of the sample must be avoided. The first step to achieve this is to identify those areas of the sample in the current reconstruction, which is why volume segmentation is needed. This segmentation was done by thresholding, aiming to isolate the sample regions whose absorption coefficient is high enough to cause artifacts in the final tomographic reconstruction described in section 7.2.7. We chose the threshold in terms of the values of the sample holder and the calibration spheres it contains in the reconstructed volume since these components are strictly necessary for extracting the exact geometry through the calibration process, as presented in [Pek+22b], and therefore they are always present. Moreover, the sample holder is made of a low-absorbing material (polyoxymethylene), so it does not interfere with the reconstruction. At the same time, the spheres are highly absorbing (aluminum) to ensure calibration success. Both values were extracted from the current reconstruction, using their known location in the reconstructed volume and creating an interval from which the threshold will be picked. However, to ensure the well-functioning of the algorithm in all scenarios, including if the sample contains parts with higher absorption than the calibration spheres or if the sample does not contain any high-absorbing material, we redefined the upper limit of this interval as the maximum between the spheres value and the maximum value within the sample region in the reconstruction, and the lower limit as the first quartile of the interval. For the threshold choice, we applied different thresholds within the interval in a decreasing fashion to the reconstruction, the final one chosen as the first one leading to a relevant segmented volume (0.0005 % of the total volume) or as the lower bound of the interval if it was reached. The result of the segmentation was a volume of the same size as the reconstruction but only containing values higher than the chosen threshold. Besides, the sample holder region was blacked out in the segmentation volume to prevent the calibration spheres from being taken as regions to be avoided.

7.2.5 Score update

In the score update step, we update the surface area parametrized by the given pose with a score. Ideally, this score should represent the sample's absorption rate when captured from the given pose. A higher sample attenuation from the given angle should result in a higher score. We separated the score update into two parts: in

the first part, we calculate the score, and in the second part, we find and update the regions on the spherical data structure corresponding to the pose.

The segmented volume is input to the score calculation procedure, which outputs an absorption score for a given pose. The score is calculated by applying a forward projection to the segmented volume with the calibrated geometry of the given pose and calculating the L0-norm on the resulting sinogram. The L0 norm effectively calculates the surface area of the non-zero pixels on the detector image. Since we are applying the forward projection on the segmented volume where solely highly absorbing parts are non-zero, this surface area will be proportional to the amount of high absorption captured from the given pose of the sample.

In the optimization loop (step **III**), the score is recalculated for all poses that were successfully attempted by our algorithm up to that point. Recalculating the scores for each pose is necessary as we obtained a new reconstruction volume with the latest successful image acquisition. The resulting scores are now comparable, and we can create a probability distribution based on these scores in the next step (see section 7.2.6). We are calculating the scores for the poses efficiently by applying a single forward projection on the reconstruction volume with the provided set of angles. The resulting sinogram contains one slice per pose, and the metric mentioned above (L0-norm) can be calculated on each slice independently.

In the score initialization step, we only set the absorption score of a pose for its surface area on the sphere with low sampling (see Fig. 7.1 **II**).

In the trajectory optimization step, we additionally set the scores of all neighbors (see Fig. 7.1 **III**, step 12). We determine the neighbors by searching for all poses on the sphere surface within a predefined distance r radians to the given pose. This update pattern can be seen on the discretized sphere in Fig. 7.1 for the red cross in step **III**, where the immediate spatial neighbors are updated with the light blue color because they fulfill the distance criterion. Applying the acquired score on the neighbors is necessary as those scores were initialized in the scout scan with a score obtained for a comparably large region. For example, when dividing the surface area of the sphere into 108 and 4032 regions in the scout scan and optimization loop, respectively, the ratio is 41. This ratio means a score from the sphere with low sampling density was set for 41 regions on the sphere with high sampling density in the score initialization step (**II**). The distance-based update step ensures that the newly obtained absorption information represented by the new score is spread on the sphere data structure in every iteration. It is helpful for the pose sampling step, where the score of a pose determines its probability of being picked for the subsequent acquisition.

7.2.6 Pose sampling

The pose sampling procedure chooses the next pose the robotic arm will approach to acquire a new image. Ideally, the sampled pose has minimal absorption out of all the remaining poses.

We start with all possible poses on the sphere data structure. We filter out the ones already attempted in previous runs of the trajectory optimization loop, regardless of successful execution.

Before introducing the probability distribution, we need to determine those poses on the sphere, for which we could not assign a score in the previous steps. These poses will not be considered in the sampling step as candidates. Three cases exist when a pose on the sphere results in an undefined state. The first case is when the pose lies in a region the robotic arm cannot reach. The second case is when the pose is inside the reachable region of the robotic arm, but the trajectory execution failed due to a hardware error. The third case is when the pose is reachable, and the trajectory was successfully executed, followed by image acquisition, but the post-processing steps failed. These steps include segmenting the circles on the image and calibrating the exact geometry of the sample on the image. Detailed information about the post-processing steps can be found in [Pek+22b].

We have illustrated the mapping of a given region on two spheres with low and high sampling in Figures 7.4 (a) and (b). It is worth noting that a pose p_i that was not reachable for the arm in the scout scan procedure is not sampled in the probability distribution defined in the next section of our implementation. In contrast, if a pose p_i was reachable, we initialize the poses p_{i1} to p_{i9} inside p_i with the score computed for p_i in the score initialization step (**II**) as illustrated in Fig. 7.4(b). This is a reasonable strategy for cases where measurement time is valuable. Another valid strategy is to initialize p_{i1} to p_{i9} with a score greater than zero, assigning sampling probabilities greater than zero for these poses in the next step. This is a more time-consuming strategy, but it is also valid as the robotic arm might still be able to reach the borders of the unreachable area represented by p_i.

In the next step, we create a discrete probability distribution where the scores for each pose are weighted with an inverse weighting scheme: Poses with a higher absorption score correspond to a lower weight in the probability distribution and vice versa. Applying an inverted weighting scheme ensures that poses with high absorption scores are less likely to be drawn from the probability distribution. Moreover, poses without a score are assigned weight 0, meaning that they will not be drawn since their probability equals 0. This happens when the neighborhood of

the pose could not be reached by the robotic arm in the scout scan, when the sphere was sampled with lower density, summing multiple poses into one pose.

We use the parameter s on the inverted weights to model a penalty on the absorption scores:

$$w_i = \frac{1}{x^s}, \tag{7.1}$$

where $s \geq 1$ is the penalty parameters and x is the absorption score. The score is raised to the power of s to place a sufficiently high penalty on highly absorbing poses. Different choices for this parameter ($s = \{5, 10, 30\}$) are displayed in Figures 7.2 (a to i) for experiments with sample number 1.

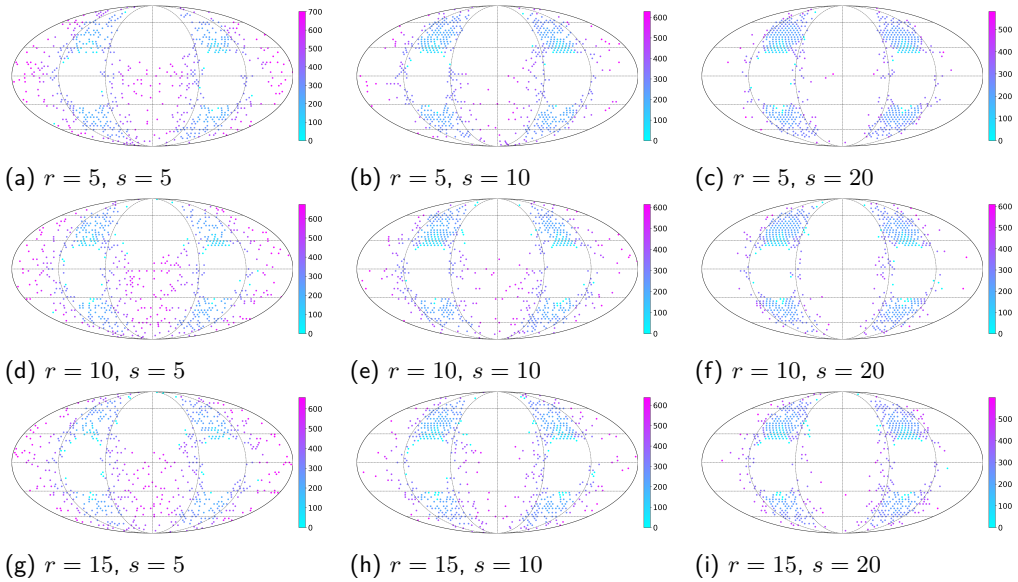
The resulting weights are used to create a discrete probability distribution with the C++ standard library component *std::discrete_distribution*. The probability of each weight element w_i is calculated with $p(i) = \frac{w_i}{S}$, where $S = \sum_i w_i$ is the sum of all weights. We sample the next valid pose from the discrete distribution with the high-entropy random number engine *std::mt19937* initialized with an entropy-based seed provided by *std::random_device*.

7.2.7 Reconstruction

We used around 725 equidistant X-ray projections for the tomographic reconstruction along three different spherical trajectories sized 720×720 pixels with a spacing of $600\ \mu m$. For the different trajectories, the number of projections varied by $+4.8\%$: the whole sphere trajectory successfully acquired and calibrated 725 images, the random trajectory 748, and the optimized trajectory 760. The remaining parameters of the reconstruction were as mentioned in section 3.3.

7.3 Experiments and results

We designed different experiments to evaluate the performance of our trajectory optimization algorithm. The first set of experiments analyzes the sensitivity of our algorithm to the hyperparameters introduced in sections 7.2.5 and 7.2.6. The second set of experiments comprehensively assesses the reconstruction image quality for two samples. We also provide a quantitative analysis of the reconstruction images compared to conventional spherical trajectories. The experiments in this section were executed in our simulation environment (see section 4.2.1).



(a) $r = 5,\ s = 5$ (b) $r = 5,\ s = 10$ (c) $r = 5,\ s = 20$

(d) $r = 10,\ s = 5$ (e) $r = 10,\ s = 10$ (f) $r = 10,\ s = 20$

(g) $r = 15,\ s = 5$ (h) $r = 15,\ s = 10$ (i) $r = 15,\ s = 20$

Fig. 7.2.: Trajectory optimization hyperparameters. We executed our trajectory optimization algorithm with different choices for the parameters r and s for sample number 1 and plotted the resulting spherical trajectories. In the three rows, we visualized the resulting trajectory when different disk radius parameters ($r = 5$, $r = 10$ and $r = 15$ radians) are applied for the neighborhood of a given pose. Poses within this neighborhood were updated with the new score if they were not yet attempted by the robotic arm. In the three columns, we visualized the resulting trajectory when different penalty parameters ($s = 5$, $s = 10$ and $s = 20$) are applied on the absorption score. s penalizes the absorption rate of the sample from the given angle by decreasing the weight of the given pose in the resulting probability distribution.

7.3.1 Trajectory optimization parameters

We have conducted experiments for the two hyperparameters of our algorithm mentioned in section 7.2. The disk radius parameter r mentioned in section 7.2.5 determines the size of the neighborhood that is affected by the new score of the given pose. A higher choice for r will apply a more significant update on the sphere surface. The score penalty parameter s mentioned in section 7.2.6 penalizes the absorption rate of the given pose. A higher choice for s will result in a lower weight of the pose in the probability distribution and hence a lower probability of being sampled for image acquisition. We used three values for each hyperparameter: $r = \{5, 10, 15\}$ and $s = \{5, 10, 20\}$. Our goal with these experiments was to analyze the sensitivity of our algorithm to different choices for the hyperparameters.

7.3.1.1. Disk radius r

Our choices for the disk radius parameters $\{5, 10, 15\}$ correspond to 8, 32, and 78 poses on the sphere with high sampling density. These numbers cover 0.18%, 0.74%, and 1.8% of the sphere surface. However, more importantly, these numbers cover 19.5%, 78%, and 190% of a single spot on the lower sampled sphere from the scout scan step. For example, by choosing $r = 15$, we would overwrite the resulting scores of two poses from the scout scan with a single measurement in one iteration of the optimization loop.

7.3.1.2. Score penalty s

The score penalty s penalizes the absorption rate of the sample from the given angle by decreasing the weight of the given pose in the resulting probability distribution. For example, our choices for $s = \{5, 10, 20\}$ decrease the weight of a pose with score 558 (higher absorption) from $2.11 \times 10^{-1}\,(s = 1)$ to $4.23 \times 10^{-4}\,(s = 5)$, $1.79 \times 10^{-7}\,(s = 10)$, and $3.20 \times 10^{-14}\,(s = 20)$ compared to a pose with score 118 (lower absorption). Considering that $p(i) = \frac{w_i}{S}$ where w_i is the weight and S is the sum of all weights, we can see that increasing the penalty parameter s yields a significant reduction in the pose's sampling probability. The scores 118 and 558 are the lowest and highest absorption score of the scout scan for sample Nr. 1. We aim to study the effect of choosing different penalty parameters s on the resulting pose distribution and find a suitable range of choices for s.

In Fig. 7.2, we plotted the results of our experiments for the parameters r and s.

7.3.2 Simulated CT measurements

We conducted six experiments in total: two samples were simulated on three trajectories: whole sphere, random, and optimized.

The three trajectories are visualized in figures 7.4 (c), (d) and (e). We generated the first trajectory by uniformly sampling poses on the sphere surface with the HEALPix library with the parameter $N_{side} = 9$, which results in 972 poses. The resulting trajectory contained fewer poses (725) than 972 as not all regions of the sphere are reachable by the robotic arm (see Fig. 7.4 (c)). We generated the second trajectory by first sampling poses on the sphere surface uniformly with a higher density ($N_{side} = 19$) and subsequently sampling poses without replacement from the

uniformly distributed poses. The resulting trajectory contained 748 poses (see Fig. 7.4 (d)). We were not able to match the number of poses in Fig. 7.4 (c) precisely as the motion planning step is executed after sampling the poses, and not all poses are reachable as mentioned earlier. The third trajectory is generated with the pipeline explained in 7.2.1. The resulting trajectory contains 760 poses (see Fig. 7.4 (e)).

The two samples we used for all experiments are displayed in Fig. 7.3. Both samples consist of objects with an internal structure and an absorber plate. We expect the absorber plate to affect reconstruction image quality badly in our measurements due to its higher attenuation coefficient than the sample objects.

The first sample is displayed in Fig. 7.3(a). It combines a cubic and a cylindrical object stacked vertically and placed next to an absorber plate. The cubic object in the bottom measures 22 x 18 x 30 mm and contains a cylindrical hole with a 7 mm diameter. The cylindrical object has a diameter of 32 mm with two cylindrical holes, with 6 mm diameter each and a height of 30 mm. Both objects total 60 mm in height. We placed the absorber plate with dimensions 4 x 30 x 50 mm next to the sample described above.

The second sample is displayed in Fig. 7.3(b). We modeled this sample as an open box that contains three sample objects, two cylinders with varying heights, and a cubic object. We placed the absorber plate with dimensions 1 x 12 x 12 mm in the center of the box. The cubic object at the top left measures 6 x 3 x 10 mm and contains a cubic hole with 1 mm padding to both sides. The cylindrical object below it measures 6 x 6 x 15 mm and contains a cylindrical hole with a 2 mm diameter. The cylindrical object to the right of the absorber measures 3 x 3 x 5 mm, and it contains a cylindrical hole with a 0.8 mm diameter.

The two samples include absorbers with different dimensions and thicknesses. The absorber in sample number 1 has a much greater volume of 6000 mm^3 compared to the absorber in sample number 2 with just 144 mm^3.

We set the absorption coefficient of the absorber plates in the sample objects to 7.0 and the rest of the samples to 1.5 for realistic modeling of the ratio between highly absorbing and less absorbing parts of a sample in a laboratory X-ray CT measurement. For reference: at an X-ray energy of 45 keV and material thickness of 3 mm, *aluminium* attenuates 30.63% of the X-rays compared to *polyethylene* with 5.87% attenuation [Web]. These values give us a ratio of 5.21 and hence are comparable to our ratio of 4.67.

For the quantitative analysis of our experiments, we have calculated a metric on the reconstructions of our experiments from section 7.3. The trajectories for these

Tab. 7.1.: Reconstruction quality statistics. Quantitative analysis of reconstruction image quality for two samples. The trajectory optimization parameters $r = 5$ and $s = 20$ were used for the underlying experiments. We calculated the metrics on the line profiles of the YZ (sample no. 1) and ZX-slices (sample no. 2) of the reconstructions (see figures 7.5 and 7.6).

	whole sphere	random	optimized
profile 1	0.000991	0.000945	0.002284 (+130.5 %)
profile 2	0.001965	0.002047	0.003631 (+84.8 %)
profile 3	0.002054	0.002099	0.004346 (+111.6 %)

(a) **Sample No. 1**: Gradient magnitude of line profiles plotted in Fig. 7.5.

	whole sphere	random	optimized
profile 1	0.000213	0.000216	0.000283 (+32.9 %)
profile 2	0.000811	0.000760	0.001466 (+80.8 %)
profile 3	0.000799	0.000766	0.001356 (+69.7 %)

(b) **Sample No. 2**: Gradient magnitude of line profiles plotted in Fig. 7.6.

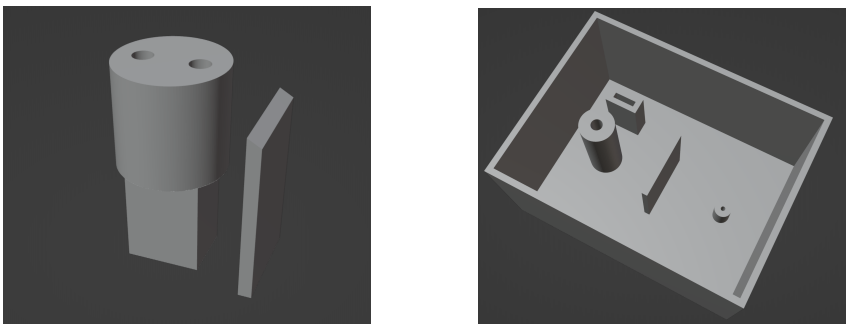
experiments are displayed in Fig. 7.4 and the qualitative results are illustrated in Figures 7.5 and 7.6. In Tables 7.1a and 7.1b, we have provided statistics on the quantitative difference in the reconstruction images between the three trajectories. We used the gradient magnitude to measure the sharpness of the line profiles on the resulting reconstruction slices (see Figures 7.6 and 7.5).

7.4 Discussion

In this section, we will discuss the results of our experiments from section 7.3, where we aimed to measure our system's performance for different choices of the hyperparameters r and s and the reconstruction image quality. We categorized the first set of experiments by the choices for the parameters r and s and fixed the sample to sample no. 1. We fixed the parameters r and s for the second set of experiments and used two different samples to evaluate our algorithm's performance on two different sample inputs.

7.4.1 Trajectory optimization parameters

We executed experiments with different choices for the parameters r and s to determine the effects on the distribution of poses on the sphere. We plotted the resulting spherical trajectories in figures 7.2 (a) to (i).

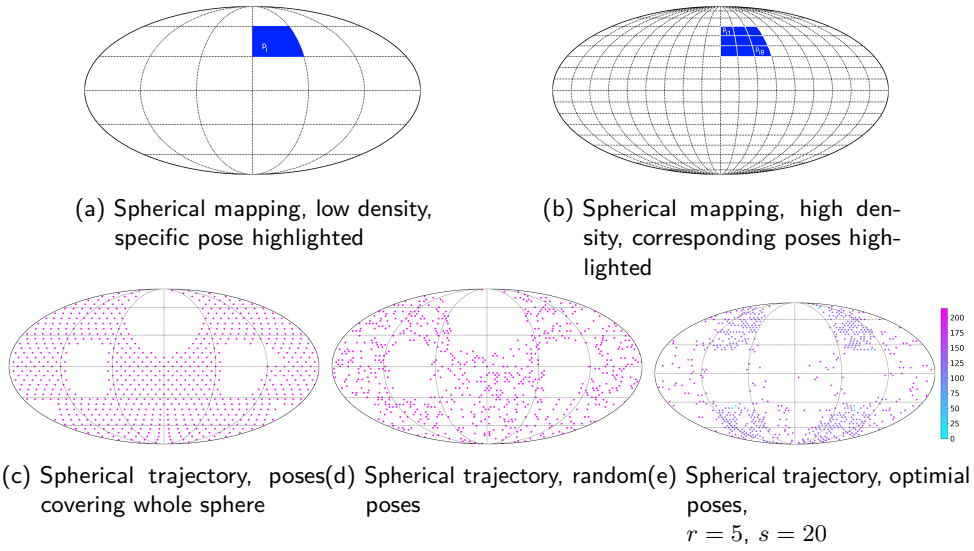


(a) Sample No. 1 (b) Sample No. 2

Fig. 7.3.: Samples. We modeled two samples, each containing an absorber plate that simulates highly absorbing parts. In (a), the sample is a combination of a cubic and a cylindrical object stacked vertically (left), placed next to an absorber plate with dimensions 4 x 30 x 50 mm. In (b), the sample is made up of an open box containing one cubic and two cylindrical objects separated by a single absorber plate with dimensions 1 x 12 x 12 mm. For both samples, we modeled holes into the cubic and cylindrical objects for evaluating reconstruction image quality.

The parameter r is the disk's radius that contains all poses in the neighborhood that are updated with the newly calculated score in the update step explained in section 7.2.5. We chose $r = \{5, 10, 15\}$ radians as update radiuses in our experiments. In section 7.3.1.1, we have mentioned that these parameters will update 0.18%, 0.74%, and 1.8% of the entire sphere surface. We can see in the rows of Fig. 7.2 that our choices for r are visible on the trajectories when fixing s by choosing a column and examining different rows for varying r. However, the difference caused by different r will not impact the reconstruction image as the distribution of points only differs in a small neighborhood when s is fixed. The re-distribution of points in a small neighborhood on a sphere does not cause significantly different angles with higher absorption.

We apply the parameter s as a penalty on the absorption score of a given pose to decrease its weight in the probability distribution. We chose $s = \{5, 10, 20\}$ as penalty parameters in our experiments. We can see in the columns of Fig. 7.2 that with increasing s, the poses on the spherical trajectory concentrate in four regions of the sphere surface. The remaining regions are sparsely populated, and poses in these regions are less likely to be sampled with increasing s. The concentration of sampled poses in particular highly absorbing regions of the sphere surface, matches our initial goal of avoiding images with high absorption. If the parameter s is chosen too high, information about certain angles of the sample will be incomplete or missing for the image reconstruction step. The information loss will lead to worse image reconstruction quality.



(a) Spherical mapping, low density, specific pose highlighted

(b) Spherical mapping, high density, corresponding poses highlighted

(c) Spherical trajectory, poses covering whole sphere

(d) Spherical trajectory, random poses

(e) Spherical trajectory, optimial poses, $r = 5$, $s = 20$

Fig. 7.4.: **Spherical trajectories**. In (a) and (b), we visualized the mapping of specific regions between sphere surfaces with low and high sampling density. In (c), (d) and (e), we plotted the spherical trajectories for the three different strategies of choosing poses on the sphere for sample no. 2: Whole sphere coverage (c), random sampling (d) and online optimization based on absorption score of previous measurements (e). In (c), (d) the spots on the sphere are colored uniformly as no score was calculated in these cases. The resulting image reconstructions for the three trajectories are displayed in Fig. 7.6.

7.4.2 Simulated CT measurements

We fixed the trajectory optimization parameters for the CT measurements to $r = 5$ and $s = 20$. These parameters improve image reconstruction quality as discussed in section 7.4.1. We used two samples (see Fig. 7.3) to evaluate the reconstruction images for image quality and sharpness.

7.4.2.1. Image quality

We can discuss our system's performance from a qualitative and quantitative perspective by examining the results in Figures 7.5 and 7.6 and Tables 7.1a and 7.1b.

We can divide our observations on the reconstruction image quality into two parts: the region around the absorber and the rest of the sample. We expect as few artifacts as possible for the region around the absorber. We expect the internal structures to be clearly visible for the remaining parts of the sample and the outer edges to be sharp.

First, we will analyze the region around the absorber. For sample no. 1, we can see in Fig. 7.5 in the first row that the optimized trajectory can avoid most of the artifacts that would otherwise be present in the ZX-slices of the reconstruction. Compared to the first row, the artifacts are not avoided as well in the YZ-slice (second row) with the optimized trajectory. However, there is still a visible difference compared to the other two trajectories. With the whole sphere (left column) and random (center column) trajectory types, the region around the absorber has artifacts in all directions. For sample no. 2, the region around the absorber has significantly fewer artifacts for the optimized trajectory (right column) than the whole sphere and random trajectory, which have artifacts in all directions. In contrast to sample no. 1, the difference in quality is significant in both slices instead of just one slice.

We will examine the outer edges and internal structures of the cylindrical and cubic objects for the remaining parts of the samples. For sample no. 1, the optimized trajectory can partly visualize the internal structure of the stacked objects next to the absorber. The other two trajectory types (left and middle column) can not visualize the internal structure in the stacked object; the object's interior is uniformly white.

For samples no. 1 and 2, we can not observe a difference in quality between the whole sphere and random trajectory for the region around the absorber and the remaining sample parts.

7.4.2.2. Image sharpness

We will evaluate reconstruction image sharpness for samples no. 1 and 2 by interpreting the line profiles plotted in Figures 7.5 and 7.6 and the corresponding statistics in Tables 7.1a and 7.1b.

In general, when looking at the line profiles for both samples, the apparent spikes are steeper and have a higher amplitude with the optimized trajectory (green lines) than the profiles of the other two trajectories (red and blue lines). There is an exception to this statement when the line profiles pass the area above the absorber on the YZ-slices for sample no. 1. Here we can see slight artifacts for the optimized trajectory when there are no visible artifacts for the other two trajectories. We suspect that acquisition angles parallel to the absorber are causing these, which is expected because these angles have a minor absorption score with the L0-norm. The minor score comes from the fact that the projected area of the absorber is smaller on the sinogram compared to an angle that captures the absorber head-on. This behavior can be fine-tuned by decreasing the penalty parameter s.

Another critical observation on the line profiles is the appearance of the samples' smaller inner structures in the optimized trajectory profiles. The profiles of the whole sphere and random trajectories do not correctly capture these regions. For example, on the overlapping plot for line profile 2 of sample no. 2 (green line), we can see the correct representation of the cylindrical hole inside the cylindrical part to the left of the absorber and the wall structure next to the sample. The whole sphere and random trajectories (blue and red lines) cannot represent this information accurately.

We can see in Tables 7.1a and 7.1b that with the optimized trajectory, our gradient magnitude metric is improved by up to 130.5% and 80.8% on the image reconstructions for sample no. 1 and 2.

In our previous publication, we used the same metric for comparing the difference in sharpness between the image reconstructions of measurements on a circular and spherical trajectory in a laboratory X-ray environment [Pek+23b]. There, the gradient magnitude improved by 50% to 80% with a spherical trajectory that covers the whole sphere as illustrated in Fig. 7.4c. This improvement was since the circular trajectory only covered a fraction of the sphere surface compared to the newly introduced spherical trajectory. With the algorithm introduced in this paper, we are optimizing the spherical trajectory from [Pek+23b]. We can conclude from the numbers in Tables 7.1a and 7.1b that with an optimized trajectory, we can improve the reconstruction image sharpness even further (up to 130.5% with sample no. 1, profile 1).

7.5 Conclusion

In this chapter, we have extensively studied the runtime optimization of spherical trajectories with the robotic sample holder, where no prior knowledge about the sample is provided. The optimization algorithm is based on an intermediate reconstruction of the measured sample. We have conducted experiments in our simulation environment with two different samples and have seen promising results. This online trajectory optimization algorithm is a promising step towards time-efficient and radiation-dose-reduced acquisitions with X-ray CT systems.

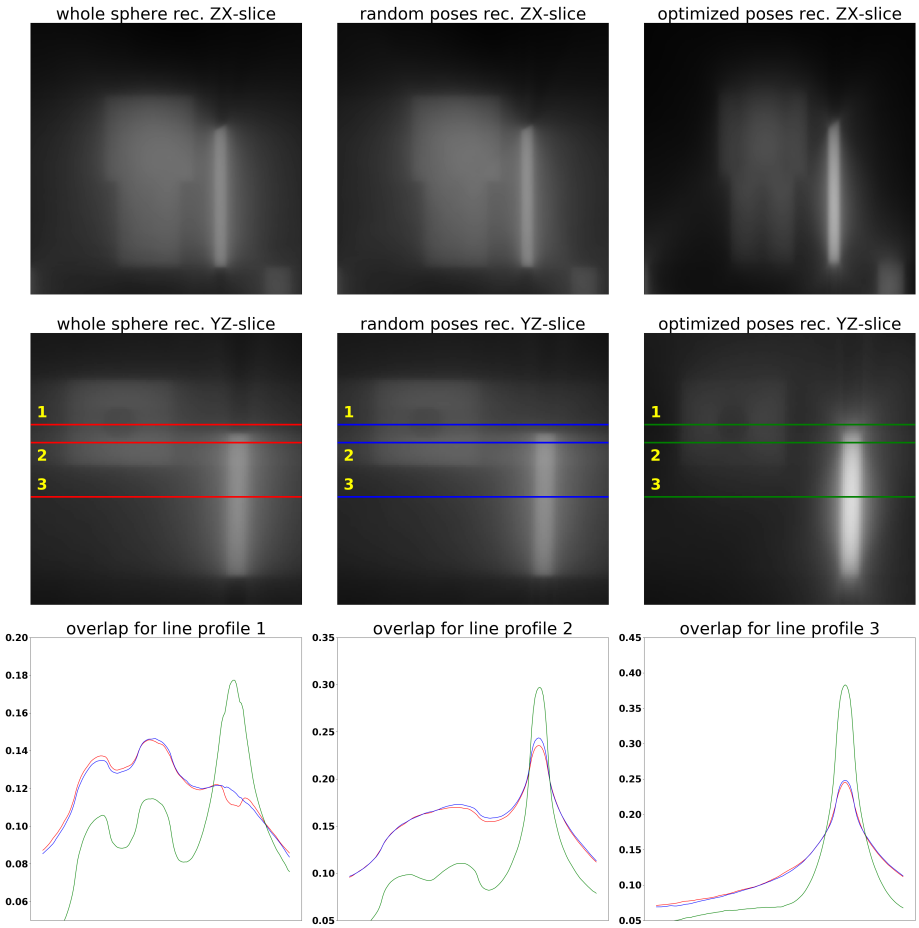


Fig. 7.5.: Experimental results. Sample No. 1. The first sample (see Fig. 7.3a) was measured on three different types of trajectories (similar to Fig. 7.4 ((c), (d) and (e))) and reconstructed with the robotic arm in the simulation environment of our robotic software package [Pek+23b]. The optimized trajectory was generated with the hyperparameters $r = 5$ and $s = 20$. The reconstruction volumes are registered and aligned with our calibration algorithm [Pek+22b]. We binned the detector images with $4 * 4$, and the reconstruction volume has dimensions 720^3. A zoom factor of 4.8x was applied to the slices to crop the region of interest. We plotted line profiles at three different positions for the YZ slices.

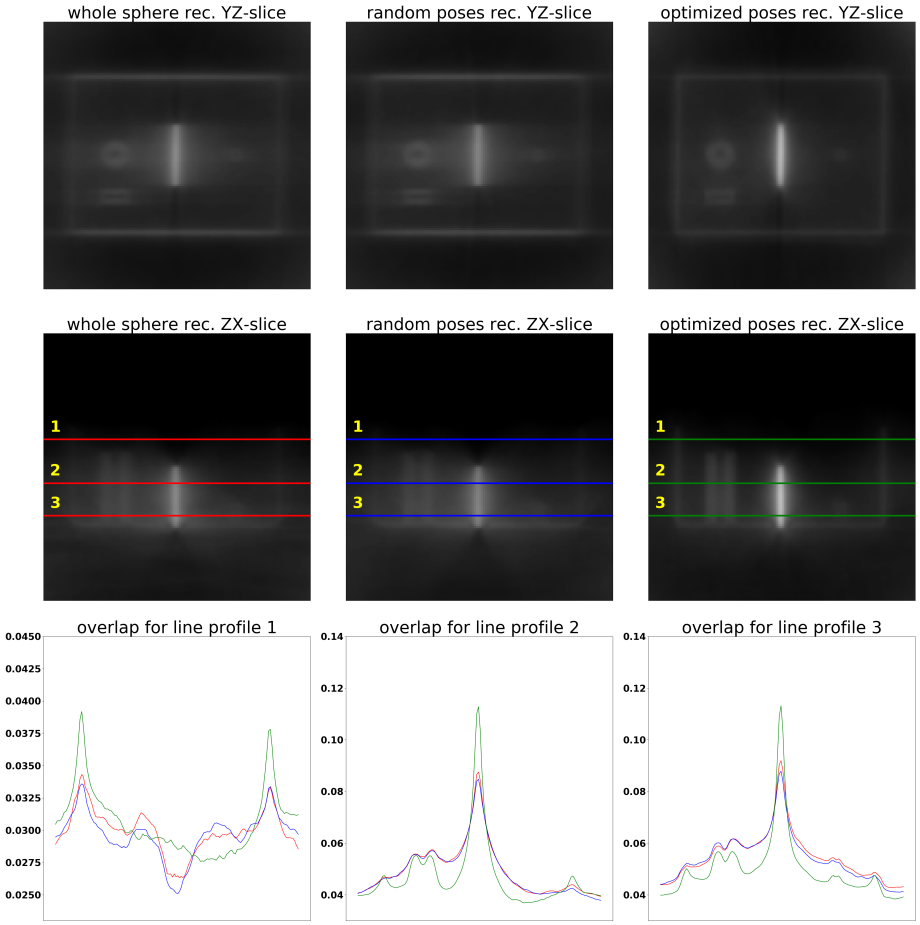


Fig. 7.6.: Experimental results. Sample No. 2. The second sample (see Fig. 7.3b) was measured on the three trajectories displayed in Fig. 7.4 ((c), (d) and (e)) and reconstructed with the robotic arm in the simulation environment of our robotic software package [Pek+23b]. The optimized trajectory was generated with the hyperparameters $r = 5$ and $s = 20$. The reconstruction volumes are registered and aligned with our calibration algorithm [Pek+22b]. We binned the detector images with $4 * 4$, and the reconstruction volume has dimensions 720^3. A zoom factor of 4.8x was applied to the slices to crop the region of interest. We plotted line profiles at three different positions for the ZX slices.

8 Conclusion

I have split the conclusion into three logical parts. I will start by providing my own view on the project with its ups and downs over the last four and a half years. Next, I will talk about what is still possible, the potential of this project. Finally, I will finish this thesis with a summary and concluding words.

8.1 Personal Conclusion

We started to work on this project in October 2018, beginning with my master's thesis, where our first goal was to create a working simulation environment for the FRANKA EMIKA Panda robotic arm. We intended to build a collision detection algorithm based on the simulation environment and implement our first software components for the motion planning pipeline in the ROS ecosystem. With the installation of the robotic arm in the X-ray CT laboratory environment and the installation and configuration of the hardware sensors, the system was ready for the first experiments. At this point, we discovered the issues of accurate sample placement with the robotic arm while executing our first experiments. Identifying the placement error and developing ideas to fix it was the first serious challenge of integrating the robotic arm as a sample holder in the laboratory since image reconstruction was unattainable without extracting the sample's exact geometry on the detector images. In the following months, we put a significant effort into designing and 3D printing an intermediate sample holder part with an embedded calibration structure that we could detect on the detector images. In my opinion, this was the most challenging part of this dissertation as we tried to fix a problem with a solution approach (sample holder and calibration structure) that allowed us to tune more than one parameter simultaneously (e.g., X-ray setup geometry, arrangement and size of calibration markers, type of calibration structure, choice of optimization algorithm). In our first journal publication, we presented our approach for the geometric calibration of the robotic sample holder [Pek+22b].

In the next step, we decided to investigate the robotic sample holder's ability to reach all possible rotations in three dimensions and hence to execute spherical

trajectories. First, we conducted literature research and encountered the various approaches to sample points on the sphere's surface uniformly. After an initial testing phase, we decided on the well-documented HEALPix library, given that it provides Python and C++ packages, and we integrated it into our software package. Second, we planned the first trajectories that covered the whole sphere with the robotic arm, and we discovered that we could not reach almost 40% of all possible rotations with the current design of the sample holder part. We also discovered that spherical trajectories were causing more occlusions with the sample on the detector images. Third, we executed the initial spherical trajectories in the laboratory, and we discovered that moving to some regions of the sphere's surface will trigger hardware reflexes of the robotic arm, interrupting trajectory execution. We designed and tested numerous gripper types with varying shapes and lengths for the intermediate sample holder part. This design process was another critical and time-consuming project phase, as many designs were possible for the gripper part. Testing these parts in the simulation environment was straightforward. However, the reduction in hardware reflexes could only be tested once we manufactured the mechanical gripper part and executed the trajectory in the laboratory. In our second journal submission, we presented our approach for maximizing the reach of the robotic sample holder while executing spherical trajectories [Pek+23b].

In this dissertation's third and last phase, we aimed to sparsely sample "valuable" spots on the spherical trajectory at runtime without prior knowledge about the sample. In the first weeks of this research phase, we invested significant time into planning a pipeline that we could implement in the remaining months of this dissertation. At the same time, the pipeline should be able to deal with complex samples and detect highly absorbing regions in it. Our initial implementation of the trajectory optimization pipeline worked with minor modifications, and we were able to submit a manuscript to the journal *Engineering Research Express*. The challenging part of the implementation was that each step of the pipeline expected the output of the preceding step as input, and hence tracking down issues in the individual steps was very challenging at the beginning.

Looking back, having a simulation environment that can handle the various samples and sample holders and that can calibrate the detector images was immensely helpful. We used the simulation intensely to implement and test the software and algorithms of the second and third manuscripts.

8.2 Future Work

In the preceding chapters, we have outlined the abilities and restrictions of the proposed system in great detail. This section will briefly discuss ideas that could improve the proposed system and algorithms.

The proposed system is currently prone to the movement of the sample and the robotic arm. For rigid samples, the movement is not an issue with this system as it is mounted to the rigid intermediate sample holder part. However, this setup can not be used for non-rigid samples as the arm moves the sample with the arm. In a new iteration of this setup, two robotics arms could be used to move the X-ray source, and the detector and the sample could be mounted statically. The vibrations caused by the robotic arm's movement in our current setup are dealt with by waiting a second after reaching a way-point on the trajectory.

New designs for the gripper types that were introduced in chapter 6 could further maximize the coverage of the sphere surface. The cylinder part of the sample holder could also be improved with a new sample mounting mechanism for mounting bigger and heavier samples.

The trajectory optimization algorithm proposed in chapter 7 could be improved in two ways. First, the existing implementation could be parallelized so that our optimization pipeline's score update and pose sampling steps are executed based on the image reconstruction and segmentation of the previous iteration. This improvement could lead to savings in execution time for the reconstruction and segmentation step and, more importantly, increased resolution of the intermediate reconstruction. Second, the pose sampling procedure could be adapted for better coverage of areas marked as unreachable when the scout scan trajectory was generated. The current strategy omits comparably large regions on the sphere surface because an unreachable pose of the sphere with low density and hence large regions covers them. With the new sampling approach, we can explore parts of these regions (e.g., borders of the unreachable poses) and increase image quality by potentially acquiring additional favorable angles in the optimization loop.

8.3 Summary

In summary, we have introduced a robotic sample holder with seven degrees of freedom for X-ray computed tomography in this dissertation. We have explored the abilities and restrictions of the given robotic arm when used in a laboratory

environment. With our unified software package, the user can efficiently simulate measurements and prototype algorithms for the robotic sample holder without entering the laboratory. Our software package also allows the execution and management of experiments safely in the laboratory environment without restricting the user to a specific hardware configuration, as it can be extended and modified with minimal effort. In our closing words, we can state that we have provided a working prototype for the robotic sample holder, that can be optimized in the future for general availability.

Bibliography

[Bal81] D H Ballard. “Generalizing the Hough transform to detect arbitrary shapes”. In: *Pattern Recognition* 13.2 (1981), pp. 111–122 (cit. on p. 68).

[BA15] Patrick Beeson and Barrett Ames. “TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics”. In: *Proceedings of the IEEE RAS Humanoids Conference*. Seoul, Korea, Nov. 2015 (cit. on p. 67).

[Bot+02] Mario Botsch, Stephan Steinberg, Stephan Bischoff, and Leif Kobbelt. “Openmesh-a generic and efficient polygon mesh data structure”. In: (2002) (cit. on p. 55).

[Bra00] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000) (cit. on p. 55).

[Cho+05] Youngbin Cho, Douglas J. Moseley, Jeffrey H. Siewerdsen, and David A. Jaffray. “Accurate technique for complete geometric calibration of cone-beam computed tomography systems”. In: *Medical Physics* 32.4 (2005), pp. 968–983 (cit. on pp. 6, 64).

[Col+14] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*. 2014. arXiv: 1404.3785 (cit. on p. 55).

[Cra05] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson-/Prentice Hall, 2005 (cit. on pp. 17, 20).

[DM98] Leonardo Dagum and Ram Menon. “OpenMP: an industry standard API for shared-memory programming”. In: 1998 (cit. on p. 55).

[FRA21] EMIKA FRANKA. *franka ros*. `https://github.com/frankaemika/franka_ros`. 2021 (cit. on p. 55).

[FRA23] EMIKA FRANKA. *franka ros*. `https://frankaemika.github.io/docs`. 2023 (cit. on p. 34).

[GSS20a] Grace J Gang, Jeffrey H Siewerdsen, and J Webster Stayman. “Non-circular CT orbit design for elimination of metal artifacts”. In: *Medical imaging 2020: physics of medical imaging*. Vol. 11312. SPIE. 2020, pp. 531–536 (cit. on p. 98).

[GSS20b] Grace J. Gang, Jeffrey H. Siewerdsen, and J. Webster Stayman. “Non-circular CT orbit design for elimination of metal artifacts”. In: *Medical Imaging 2020: Physics of Medical Imaging*. Ed. by Guang-Hong Chen and Hilde Bosmans. Vol. 11312. International Society for Optics and Photonics. SPIE, 2020, p. 1131227 (cit. on p. 80).

[Gor+05] K. M. Gorski, E. Hivon, A. J. Banday, et al. “HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere”. In: *The Astrophysical Journal* 622 (2 2005), pp. 759–771 (cit. on pp. 55, 84).

[HJL21] Per Christian Hansen, Jakob Jørgensen, and William R. B. Lionheart. *Computed Tomography: Algorithms, Insight, and Just Enough Theory*. Ed. by Per Christian Hansen, Jakob Jørgensen, and William R. B. Lionheart. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2021. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611976670` (cit. on pp. 25, 26).

[HS95] R. H. Hardin and N. J.A. Sloane. “New spherical designs in three and four dimensions”. In: *IEEE International Symposium on Information Theory - Proceedings* 441 (1995), p. 181 (cit. on p. 83).

[Hat+22] S Hatamikia, A Biguri, G Herl, et al. “Source-detector trajectory optimization in cone-beam computed tomography: a comprehensive review on today’s state-of-the-art”. In: *Physics in Medicine & Biology* 67 (16 Aug. 2022), 16TR03 (cit. on p. 80).

[Her+21] Gabriel Herl, Jochen Hiller, Mareike Thies, et al. “Task-Specific Trajectory Optimisation for Twin-Robotic X-Ray Tomography”. In: *IEEE Transactions on Computational Imaging* 7 (2021), pp. 894–907 (cit. on pp. 5, 63, 80).

[Her+14] Andreas Hermann, Florian Drews, Joerg Bauer, et al. “Unified GPU voxel collision detection for mobile manipulation planning”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 4154–4160 (cit. on p. 49).

[Hun07] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95 (cit. on p. 55).

[Jos82] Peter M. Joseph. “An Improved Algorithm for Reprojecting Rays through Pixel Images”. In: *IEEE Transactions on Medical Imaging* 1.3 (1982), pp. 192–196 (cit. on p. 30).

[KKM14] Zubair Khalid, Rodney A. Kennedy, and Jason D. McEwen. “An optimal-dimensionality sampling scheme on the sphere with fast spherical harmonic transforms”. In: *IEEE Transactions on Signal Processing* 62 (17 2014), pp. 4597–4610 (cit. on p. 83).

[KH04] Nathan Koenig and Andrew Howard. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2149–2154 (cit. on pp. 38, 39).

[LHH19] Peter Landstorfer, Gabriel Herl, and Jochen Hiller. “Investigation of non-circular scanning trajectories in robot-based industrial X-ray computed tomography of multi-material objects”. In: *ICINCO 2019 - Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics* 2.Icinco (2019), pp. 518–522 (cit. on pp. 5, 62).

[LHF19] Tobias Lasser, Maximilian Hornung, and David Frank. “elsa - an elegant framework for tomographic reconstruction”. In: *15th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*. Ed. by Samuel Matej and Scott D. Metzler. Vol. 11072. International Society for Optics and Photonics. SPIE, 2019, pp. 570–573 (cit. on pp. 30, 38, 55).

[LaV06] Steven M. LaValle. *Planning Algorithms*. New York, USA: Cambridge University, 2006 (cit. on pp. 7, 10).

[LZL10] Xinhua Li, Da Zhang, and Bob Liu. “A generic geometric calibration method for tomographic imaging systems with flat-panel detectors - A detailed implementation guide”. In: *Medical Physics* 37.7 (2010), pp. 3844–3854 (cit. on pp. 6, 64).

[Ma+22] Yiqun Ma, Grace J. Gang, Tina Ehtiati, et al. “Non-circular CBCT orbit design and realization on a clinical robotic C-arm for metal artifact reduction”. In: *Medical Imaging 2022: Image-Guided Procedures, Robotic Interventions, and Modeling*. Ed. by Cristian A. Linte and Jeffrey H. Siewerdsen. Vol. 12034. International Society for Optics and Photonics. SPIE, 2022, 120340A (cit. on p. 80).

[Pek] Erdal Pekel. *personal blog [online]*. `https://erdalpekel.de/` (cit. on pp. 49, 58).

[Pek21] Erdal Pekel. *Robotic Sample Holder*. `https://gitlab.lrz.de/IP/robotic-sample-holder/robotic-sample-holder`. 2021 (cit. on pp. 39, 66).

[Pek+23a] Erdal Pekel, Martin Dierolf, Franz Pfeiffer, and Tobias Lasser. “Spherical acquisition trajectories for X-ray Computed Tomography with a robotic sample holder”. In: *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine (Fully 3D)*. 2023 (cit. on pp. 33, 79, 93, 95).

[Pek+23b] Erdal Pekel, Martin Dierolf, Franz Pfeiffer, and Tobias Lasser. “Spherical acquisition trajectories for x-ray computed tomography with a robotic sample holder”. In: *Engineering Research Express* 5.3 (Aug. 2023), p. 035045 (cit. on pp. 34–37, 54, 79, 93–96, 98, 99, 101, 114–116, 118).

[Pek+23c] Erdal Pekel, María Lancho Lavilla, Franz Pfeiffer, and Tobias Lasser. “Runtime optimization of acquisition trajectories for x-ray computed tomography with a robotic sample holder”. In: *Engineering Research Express* 5.4 (Nov. 2023), p. 045058 (cit. on p. 97).

[Pek+22a] Erdal Pekel, Florian Schaff, Martin Dierolf, Franz Pfeiffer, and Tobias Lasser. “Geometric calibration of seven degree of freedom robotic sample holder for x-ray CT”. In: *7th International Conference on Image Formation in X-Ray Computed Tomography*. Ed. by Joseph Webster Stayman. Vol. 12304. International Society for Optics and Photonics. SPIE, 2022, p. 123042L (cit. on p. 61).

[Pek+22b] Erdal Pekel, Florian Schaff, Martin Dierolf, Franz Pfeiffer, and Tobias Lasser. "X-ray computed tomography with seven degree of freedom robotic sample holder". In: *Engineering Research Express* 4.3 (Aug. 2022), p. 035022 (cit. on pp. 33, 35–37, 48, 54, 61, 66, 71, 72, 77–82, 85, 89, 98, 99, 102, 103, 105, 115–117).

[Qui+09] Morgan Quigley, Ken Conley, Brian Gerkey, et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5 (cit. on p. 55).

[Raf19] Boaz Rafaely. *Fundamentals of Spherical Array Processing*. Vol. 16. Springer International Publishing, 2019, E1–E3 (cit. on p. 83).

[Rey+23] Tess Reynolds, Yiqun Ma, Tianyu Wang, et al. "Revealing pelvic structures in the presence of metal hip prostheses via non-circular CBCT orbits". In: *Medical Imaging 2023: Image-Guided Procedures, Robotic Interventions, and Modeling*. Ed. by Cristian A. Linte and Jeffrey H. Siewerdsen. Vol. 12466. International Society for Optics and Photonics. SPIE, 2023, 124660Y (cit. on p. 80).

[Rob+09] Normand Robert, Kristina N. Watt, Xinying Wang, and James G. Mainprize. "The geometric calibration of cone-beam systems with arbitrary geometry". In: *Physics in Medicine and Biology* 54.24 (2009), pp. 7239–7261 (cit. on pp. 6, 64).

[SK97] E. B. Saff and A. B.J. Kuijlaars. "Distributing many points on a sphere". In: *Mathematical Intelligencer* 19 (1 1997), pp. 5–11 (cit. on p. 83).

[SGS10] J. E. Stone, D. Gohara, and G. Shi. "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems". In: *Computing in Science Engineering* 12.3 (May 2010), pp. 66–73 (cit. on p. 55).

[SC] Ioan A. Sucan and Sachin Chitta. *MoveIt: [online]*. `http://moveit.ros.org/` (cit. on p. 55).

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272 (cit. on p. 55).

[Web] G. Weber. *X-Ray attenuation & absorption calculator [online]*. `https://web-docs.gsi.de/~stoe_exp/web_programs/x_ray_absorption/index.php` (cit. on p. 109).

[Wie+18] Matthias Wieczorek, Florian Schaff, Christoph Jud, et al. "Brain connectivity exposed by anisotropic X-ray dark-field tomography". In: *Scientific reports* 8.1 (2018), pp. 1–6 (cit. on p. 62).

[Wie+16] Matthias Wieczorek, Florian Schaff, Franz Pfeiffer, and Tobias Lasser. "Anisotropic x-ray dark-field tomography: A continuous model and its discretization". In: *Physical review letters* 117.15 (2016), p. 158101 (cit. on p. 62).

[Wil79] Ralph A Willoughby. "Solutions of ill-posed problems (AN Tikhonov and VY Arsenin)". In: *SIAM Review* 21.2 (1979), p. 266 (cit. on p. 30).

[Wu+20] P Wu, N Sheth, A Sisniega, et al. “C-arm orbits for metal artifact avoidance (MAA) in cone-beam CT”. In: *Physics in Medicine & Biology* 65.16 (2020), p. 165012 (cit. on p. 98).

[Zie+20] Anya Ziertmann, Peter Jahnke, Stefan Kerscher, Michael Koch, and Wolfgang Holub. “Robot guided Computed Tomography—Production Monitoring in Automotive Industry 4.0”. In: *Journal of the Japan Society for Precision Engineering* 86.5 (2020), pp. 316–322 (cit. on pp. 5, 62).

[Zon+19] Andrea Zonca, Leo Singer, Daniel Lenz, et al. “healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in Python”. In: *Journal of Open Source Software* 4.35 (Mar. 2019), p. 1298 (cit. on p. 55).

Webpages

[@Dai22] Wei Dai. *Crypto++® Library 8.7*. 2022. URL: https://www.cryptopp.com/ (cit. on p. 55).

[@EMI20] FRANKA EMIKA. *PANDA DATASHEET*. 2020. URL: https://s3-eu-central-1.amazonaws.com/franka-de-uploads/uploads/2019/04/Datasheet.pdf (cit. on pp. 5, 33, 34, 63).

[@Hea21] SIEMENS Healthineers. *Siemens Artis Zeego Eco*. 2021. URL: https://www.siemens-healthineers.com/refurbished-systems-medical-imaging-and-therapy/ecoline-refurbished-systems/angiography-ecoline/artis-zeego-eco (cit. on pp. 5, 62, 63).

[@Int23a] Intel. *Intel Realsense Depth Camera D435*. 2023. URL: https://www.intelrealsense.com/depth-camera-d435/ (visited on Apr. 6, 2023) (cit. on pp. 37, 38).

[@Int23b] Intel. *Intel realsense-ros GitHub repository*. 2023. URL: https://github.com/IntelRealSense/realsense-ros (cit. on p. 37).

[@KUK21] KUKA. *KUKA KR 90 R3100 extra HA DATASHEET*. 2021. URL: https://www.kuka.com/-/media/kuka-downloads/imported/6b77eecacfe542d3b736af377562ecaa/0000208694_de.pdf (visited on July 7, 2022) (cit. on p. 63).

List of Figures

List of Tables

List of Listings

Appendix

A

A.1 Journal publications

- *E. Pekel, F. Schaff, M. Dierolf, F. Pfeiffer, T. Lasser*. **X-ray computed tomography with seven degree of freedom robotic sample holder**. Engineering Research Express 4, 2022
- *E. Pekel, M. Dierolf, F. Pfeiffer, T. Lasser*. **Spherical acquisition trajectories for X-ray Computed Tomography with a robotic sample holder**. Engineering Research Express 5, 2023
- *E. Pekel, M. Lancho Lavilla, F. Pfeiffer, T. Lasser*. **Runtime optimization of acquisition trajectories for X-ray computed tomography with a robotic sample holder**. Engineering Research Express, 2023

A.2 Conference Publications

- *E. Pekel, M. Dierolf, F. Pfeiffer, T. Lasser*. **X-ray Computed Tomography with a Robotic Sample Holder**. International Conference on Image Formation in X-ray Computed Tomography (CT Meeting), Regensburg, Germany, August, 2020
- *E. Pekel, F. Schaff, M. Dierolf, F. Pfeiffer, T. Lasser*. **Geometric calibration of seven degree of freedom Robotic Sample Holder for X-ray CT**. International Conference on Image Formation in X-ray Computed Tomography (CT Meeting), Baltimore, USA, June, 2022
- *E. Pekel, M. Dierolf, F. Pfeiffer, T. Lasser*. **Spherical acquisition trajectories for X-ray Computed Tomography with a robotic sample holder**. Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine (Fully 3D), Stony Brook, USA, July, 2023

A.3 ArXiv Pre-Prints

- *E. Pekel, M. Dierolf, F. Pfeiffer, T. Lasser*. **Spherical acquisition trajectories for X-ray Computed Tomography with a robotic sample holder**. ArXiv, 2023
- *E. Pekel, M. Lancho Lavilla, F. Pfeiffer, T. Lasser*. **Runtime optimization of acquisition trajectories for X-ray computed tomography with a robotic sample holder**. ArXiv, 2023