Faculty of Electrical Engineering, Computer Science and
Information Technology Osijek,
J. J. Strossmayer University of Osijek, Croatia

Valentin Šimundić

Multi-Contact Door Opening by a Robotic Arm Using an RGB-D Camera, a Force-Torque Sensor and a Tactile Sensor

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Science, scientific area of Technical Sciences,
scientific field of Computer Science at J. J. Strossmayer University of Osijek

19 February 2026
Osijek, Croatia

Multi-Contact Door Opening by a Robotic Arm Using an RGB-D Camera, a Force-Torque Sensor and a Tactile Sensor

Višekontaktno otvaranje vrata robotskom rukom pomoću RGB-D kamere, senzora sile i momenta te senzora dodira

Valentin Šimundić, © February 2026

MENTOR

prof. Robert Cupec, Ph.D., Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, J. J. Strossmayer University of Osijek, Croatia

Acknowledgements

Although my name appears on this thesis, it would not have been possible without the support and guidance of many people. I would like to thank some of those who helped me reach this point.

First and foremost, I would like to thank my mentor, Professor Robert Cupec. You have been an extraordinary mentor, and I am deeply grateful for your guidance through this fun, but sometimes challenging, process. I have learned an incredible amount from you, and I continue to learn every day. I have truly enjoyed working with you, and I hope this is only the beginning of our collaboration.

Thank you, Lukrecia Vulić, my dear friend who became my great love and my strongest support. Thank you for your patience and understanding during the times when I was constantly working or thinking about work. Thank you for always being there for me, listening, advising, and caring. You have made each new day much more special.

I would also like to thank Matej Džijan, my office mate and, more importantly, one of my closest friends. You have always been there to help me, encourage me, and keep me from doubting myself. I am grateful for all our antics and silliness, and I hope we never become too serious. Thank you, Petra Pejić, for always taking the time to answer my endless questions, for your advice, and for encouraging me to step out of my comfort zone through your example. Thank you, Jana Dukić and Andrej Bošnjak, for being great colleagues and friends, and for all the lunches and coffee breaks we shared.

I would like to acknowledge the Faculty of Electrical Engineering, Computer Science and Information Technology Osijek for giving me the opportunity to work in academia, and the Croatian Science Foundation for supporting my work under the projects IP-2019-04-6819 and DOK-2021-02.

I would also like to express my gratitude to my friends outside of work, whose friendship I deeply cherish. Thank you, Marko Greganić, Montano Fabijančić, Ivan Visković, and Ivan Smoljo, for all the jokes, laughter, and much-needed distractions from work. Thank you as well to Marko Grubeša, Marija Mihalina, and Ivan Babel for all the wonderful times we have shared, even if we do not see each other as often as we should.

Last, but certainly not least, I wish to express my deepest gratitude to my parents, Jasna and Tomislav, and my sister, Ivana. Without your love, support, and encouragement, I would not have the beautiful life I have today.

Valentin Šimundić
February 2026

Abstract

Autonomous service robots are increasingly deployed in human-centered environments such as homes, hospitals, and workplaces. Unlike structured industrial settings, these environments are cluttered, dynamic, and often unpredictable, requiring robots to operate safely around people while handling uncertainty in object pose, geometry, and state. To be useful in everyday tasks, a robot must not only perceive its surroundings but also interact with objects reliably. Many practical tasks therefore depend on manipulating articulated objects, which consist of multiple parts connected by translational or rotational joints. Doors and drawers are among the most common examples, and the ability to perceive their location and state and to manipulate them reliably is a prerequisite for many other activities. Interacting with articulated objects remains challenging because it requires accurate scene understanding as well as robust physical interaction under sensing noise, calibration errors, and model mismatches.

This thesis addresses these challenges through an integrated pipeline for detecting doors and drawers from human demonstrations, planning multi-contact motions for opening handleless cabinet doors, and improving real-world execution through multimodal failure detection and recovery. In the training phase, the proposed perception method reconstructs articulated object models from RGB-D sequences of human demonstrations and inserts them into an environment map. At run time, the system estimates the state of a previously detected door or drawer from a single RGB-D observation. Experiments on real-world RGB-D data demonstrate robust detection and accurate state estimation across multiple furniture types.

To address robust manipulation of handleless doors, the thesis formulates door opening as a multi-contact path planning problem. The proposed approach searches over feasible end-effector configurations distributed over the door surface to generate collision-free and kinematically feasible opening paths. Simulation and physical trials show that the multi-contact approach increases the number of feasible solutions when compared to single-contact methods and enables reliable door opening across a range of cabinet poses and configurations.

To improve robustness in real-world execution, this thesis proposes a door-opening framework that integrates visual, force, and tactile feedback to detect missed contact, contact loss, and collisions and to recover from these failures. A key component of the framework is a correction method that updates camera parameters based on unsuccessful manipulation attempts. The refined parameters are then used to improve the reconstructed cabinet model and enable more reliable door opening in subsequent attempts. Real-world experiments show that the proposed failure detection and correction mechanisms reduce failed actions over time and improve execution reliability.

Sažetak

Autonomni servisni roboti sve se češće primjenjuju u svakodnevnim okruženjima, poput domova, bolnica i radnih mjesta. Za razliku od strukturiranih industrijskih okruženja, svakodnevna okruženja sadrže mnogo prepreka i objekata te su dinamična i često nepredvidiva. Zbog toga roboti moraju sigurno djelovati u blizini ljudi, uz istodobno suočavanje s pogreškama u pogledu položaja, geometrije i stanja objekata. Kako bi bili korisni u svakodnevnim zadacima, roboti moraju ne samo opažati okolinu, nego i pouzdano fizički interagirati s objektima. Mnogi zadaci stoga ovise o manipulaciji artikuliranim objektima, koji se sastoje od više dijelova povezanih translacijskim ili rotacijskim zglobovima. Vrata i ladice među najčešćim su primjerima, a sposobnost procjene njihova položaja i stanja te pouzdane manipulacije preduvjet je za mnoge druge aktivnosti. Interakcija s artikuliranim objektima i dalje je zahtjevna jer traži precizno razumijevanje scene, kao i robusnu fizičku interakciju u prisutnosti šuma mjerenja, pogrešaka kalibracije i odstupanja između modela i stvarnog svijeta.

Ovaj doktorski rad pristupa rješavanju navedenih izazova kroz integrirani sustav koji obuhvaća detekciju vrata i ladica iz ljudskih demonstracija, planiranje putanja za višekontaktno otvaranje vrata ormarića bez ručki te povećanje uspješnosti otvaranja vrata u stvarnim uvjetima pomoću detekcije i oporavka od pogrešaka na temelju podataka s više senzora. U fazi učenja, predložena metoda detekcije rekonstruira modele artikuliranih objekata iz niza RGB-D slika ljudskih demonstracija i ugrađuje ih u kartu okruženja. Tijekom rada, sustav procjenjuje stanje prethodno detektiranih vrata i ladica na temelju jedne RGB-D slike. Pokusi na stvarnim RGB-D podacima pokazuju robusnu detekciju i točnu procjenu stanja na više tipova namještaja.

U svrhu robusne manipulacije vratima bez ručki, u ovom radu otvaranje vrata predstavljeno je kao problem planiranja putanje s više kontaktnih točaka. Predloženi pristup pretražuje izvedive konfiguracije krajnjeg efektora raspoređene po površini vrata kako bi generirao putanje otvaranja bez kolizija i izvedive s obzirom na kinematička ograničenja robota. Pokusi u simulacijama i na stvarnom robotu pokazuju da višekontaktni pristup, u odnosu na pristupe koji se oslanjaju na jedan kontakt, povećava broj mogućih rješenja te omogućuje pouzdano otvaranje vrata za različite položaje i konfiguracije ormarića.

Kako bi se povećala robusnost izvedbe u stvarnim uvjetima, u ovom se radu predlaže okvir za otvaranje vrata koji integrira informacije kamere, sile i dodira radi detekcije sudara, promašaja vrata i gubitka kontakta te oporavka od takvih pogrešaka. Ključni dio okvira predstavlja metoda umjeravanja koja ažurira parametre kamere na temelju neuspješnih pokušaja manipulacije. Tako dobiveni parametri koriste se za poboljšanje rekonstruiranog modela ormarića i povećanje pouzdanosti otvaranja u sljedećim pokušajima. Pokusi u stvarnim uvjetima pokazuju da predloženi mehanizmi detekcije pogrešaka i umjeravanja s vremenom smanjuju broj neuspješnih pokušaja i povećavaju pouzdanost otvaranja vrata.

Contents

Abstract **III**

Sažetak **IV**

List of Figures **VII**

List of Tables **XI**

1 Introduction **1**
1.1 Contributions 2
1.2 Organization of the Thesis 2

2 Detection of Doors and Drawers from Human Demonstration **3**
2.1 Related Work 4
2.1.1 Articulated Object Detection 4
2.1.2 Estimation of Kinematic Parameters 5
2.1.3 Door and Drawer Detection 6
2.1.4 Learning from Demonstration 7
2.2 Augmented Environment Map 7
2.2.1 Door Model 7
2.2.2 Drawer Model 8
2.3 Teaching by Human Demonstration 9
2.3.1 Clustering 10
2.3.2 Moving Part Hypotheses 11
2.3.3 Door/Drawer Hypotheses 13
2.3.4 Hypothesis Evaluation and Selection 18
2.4 State Estimation 21
2.5 Experimental Evaluation 22
2.5.1 Door/Drawer Detection 22
2.5.2 State Estimation 23
2.6 Conclusion 25

3 Multi-Contact Door Opening **27**
3.1 Related Work 27
3.1.1 Motion Planning 28
3.1.2 Door Opening Manipulation 29
3.2 Problem Definition 30
3.2.1 Furniture Model 30
3.2.2 Path Planning Problem 32
3.3 Method 34
3.3.1 Planning Door-Opening Paths 34
3.3.2 Pre-Computation of the Contact Pose Graph 36

3.3.3 Collision Detection 37
3.3.4 Node Sampling 40
3.3.5 Computational Complexity 41
3.4 Experimental Setup 42
3.4.1 Simulation Experiments 42
3.4.2 Generating Cabinets in Simulation 43
3.4.3 Generating Cabinet Poses in the Real-World Experiments 44
3.5 Results 45
3.5.1 TSR-Based Single-Contact Method 48
3.5.2 Single-Contact Simulation Experiment 49
3.5.3 Multi-Contact Parameter Selection 50
3.5.4 Multi-Contact Simulation Experiment 51
3.5.5 Handle-to-Multi-Contact Method 52
3.5.6 Real-World Experiments 53
3.5.7 Execution Time Analysis 56
3.6 Conclusion 57

4 Visuo-Force-Tactile Integration for Door Opening **59**
4.1 Related Work 60
4.1.1 Visuo-Force-Tactile Sensor Fusion 61
4.1.2 Manipulation with Failure Recovery 62
4.2 Door-Opening Framework 62
4.2.1 Door Detection Module 63
4.2.2 Correction Module 64
4.2.3 Path Planning Module 64
4.2.4 Path Execution Module 64
4.2.5 Recapture Module 65
4.3 Correction of Camera Parameters 65
4.3.1 Correction Model 65
4.3.2 Correction of the Environment Model 67
4.3.3 Contact Constraints 67
4.3.4 Optimal Correction 71
4.4 Implementation 73
4.4.1 Door Model 73
4.4.2 Path Planning 73
4.4.3 Correction 76
4.5 Experiments 76
4.5.1 Simulation Experiments 77
4.5.2 Real-World Experiments 79
4.6 Results 80
4.6.1 Results of Simulation Experiments 80
4.6.2 Results of Real-World Experiments 82
4.7 Conclusion 86

5 Conclusion **88**

A Correction of Environment Vertices **90**

B Correction of Environment Plane Parameters **92**

Bibliography **94**

List of Figures

2.1 A human demonstrates to a robot the movement of a door. The robot learns where the door is located and its direction of opening. 4
2.2 An example of an augmented environment map showing the locations of different pieces of furniture in the Institut de Robòtica i Informàtica Industrial (IRI), Barcelona. 8
2.3 Door model 9
2.4 Drawer model 9
2.5 Example of human removal from depth images using TensorMask. The left image shows the segmented human, while the right image displays the depth map with the corresponding pixels removed. 10
2.6 Illustration of subsampling of an RGB-D image. The depth image (right) is overlaid with a grid, where representative points are selected within each cell based on surface normal consistency. Cell (a) contains a single planar surface and is represented by one point, while cell (b) includes multiple surfaces at an edge and is represented by multiple points. 12
2.7 Computation of the dominant displacement vector from two point clouds within a predefined temporal window. Displacement vectors obtained via nearest-neighbor matching are shown in pink, with the dominant displacement vector highlighted in blue. Vectors that are sufficiently long and are aligned with the dominant direction are marked with green endpoints, while misaligned vectors are marked with blue endpoints. 13
2.8 The front face of an MPH whose boundary is approximated by line segments and two bounding boxes (dashed lines). The blue bounding box is aligned with the blue line segment and the green bounding box is aligned with the green line segment. 14
2.9 Input image (left) and a moving part hypothesis denoted by red lines (right). Colored regions in the right image represent detected planar patches. 14
2.10 Illustration of the moving part plane intersection (MPPI). A plane intersection line (blue) is first computed from the intersection of two moving surfaces (light green and light blue). Then, a plane Π is defined such that it is perpendicular to the z-axis hypothesis and passes through the origin of the camera reference frame S_C. The MPPI is obtained as the intersection point between the line and the plane Π. 16
2.11 Example of a robot estimating the state of an articulated object. The left image shows the robot capturing the test object in a specific state, while the right image presents the corresponding view from the robot's camera. 22
2.12 Door/drawer detection results for the test objects. The ground truth data is shown with a green bounding box, while the detected parts are shown with red bounding boxes. 24
2.13 Examples of correct (green) and incorrect (red) state estimates. 26

3.1 Real-world experimental setup featuring a cabinet with handleless doors and a push-latch opening mechanism, operated by the Universal Robots UR5 robotic arm equipped with the Robotiq 3-Finger Gripper. The ArUco board was used to ensure proper hand–eye calibration. 28
3.2 Reference frames of the proposed furniture door model. The frame S_F (purple) is fixed to the furniture, S_D (green) is attached to the back corner of the door panel, and $S_{A'}$ is obtained by rotating the reference frame S_A around its z-axis. 31
3.3 Collision model of the Robotiq 3-Finger Gripper, with the tool contact surface marked in green (left) and an approximation with a set of spheres (right). The reference tool contact point is represented by a red circle. . . . 33
3.4 Exploration graph created by the proposed algorithm. 37
3.5 Maximal expected and actual door panel. 38
3.6 Illustration of the UR5 forearm modeled with spherocylinders for efficient collision detection. The 3D visualization on the left shows the spherocylinders and the gripper mesh, with the remaining links represented as red lines. The image on the right displays the physical UR5 robot with the spherocylinders overlaid on the corresponding links. 39
3.7 Visualization of the original and inflated collision mesh models of the Robotiq 3-Finger Gripper. The inflated model is shown semi-transparent to highlight the added safety margin around the original model. 40
3.8 An illustration showing the pose space of the cabinet in the robot's environment. The figure illustrates the possible positions of the RF S_D during the entire door-opening movement. At all times, the origin of S_D must remain within the radius of 0.9 m. In addition, the line segment that connects S_A with the point S_D must remain outside the circular exclusion zone with a radius of 0.3 m. The defined cabinet pose space prevents contact between the generated cabinet and the second link of the robot (highlighted in orange) and ensures that the gripper can reach any potential contact point on the back of the door panel. 44
3.9 Illustration of the initial door state. The figure shows the cabinet door in its zero-state and an initial slightly open configuration caused by the extension of the push-latch mechanism. The push-latch mechanism, highlighted in green, has an extension length λ, while the dimensions of the door are defined by l_y and l_z. 45
3.10 The image showcases the defined cabinet pose space in the real-world experiments. The gray area represents the table surface, on which the robot is mounted. Within this space, the pose of reference frame S_A is randomly generated, confined to the yellow region. 46
3.11 Illustration of failure types observed in the simulation experiments. The failures are grouped into four categories: (a) contact dynamics issues, (b) missed approach paths, (c) rotational collisions and (d) panel collisions (highlighted in red). For each category, the corresponding planned motion and the resulting failure outcome are shown in adjacent columns. 47
3.12 A graph showcasing the categorization of failed instances for each method in the simulation. 54

3.13 Examples of simulation experiments for each tested method. The top row shows the robot executing the door-opening task using the TSR, single-contact (SC), multi-contact (MC) and handle-to-multi-contact (HMC) approaches, respectively. The bottom row illustrates the corresponding contact locations on the door surface. The TSR method interacts with the door handle and maintains a single, fixed contact point. Similarly, the SC method maintains a consistent contact point on the back surface of the door throughout the motion. In contrast, the MC method changes the contact point during execution, enabling the gripper to slide along the door's back surface. The HMC method combines handle-based opening with the multi-contact approach. Contact points are shown as blue circles. 55
3.14 Cabinets used in the real-world experiments. 56
3.15 A sequence of images showing the multi-contact door opening in a real-world experiment. The images from 1 to 8 refer to the different states of door opening in a chronological order. 57

4.1 Overview of the equipment used in the proposed door-opening system, including a LiDAR camera, force-torque sensor, and tactile sensor mounted on the robotic end-effector. 60
4.2 Overview of the proposed door-opening framework structured as a finite-state machine. Each state corresponds to a component responsible for a specific task, including door detection, planning, environment model creation and update, path execution, correction, and recapture. Transitions between the modules are triggered by visual, force or tactile feedback. Regular transitions (black) are used to control the nominal sequence, while error transitions (red) handle various failure cases, such as missed contacts, collisions or contact losses. 63
4.3 Vertex-face (a) and edge-edge (b) contact. The robot tool is painted in green and the environment object in white and orange. The orange color indicates the faces adjacent to the contact face f^e. In the case of vertex-face contact (a), the vertex of the tool, shown with a red circle, touches the white face of the environment object. In the case of edge-edge contact, the tool edge represented by the red line touches the edge of the environment object represented by the blue line. 68
4.4 Illustration of the edge-edge intersection constraint. The cross product of the tool edge (red) and the environment edge (magenta) defines a separating plane Π_k (blue) with normal n_k. The contact is valid only if the local geometries of the tool (green) and the environment (white) occupy opposite sides of this separating plane. 70
4.5 The tool (green) misses the target object. The real target object is shown with black lines, while the target object model used for action planning is shown with dashed blue lines. The arrows show the planned movement of the tool. 71
4.6 Reference frames of the door model. S_A denotes the door axis frame, while S_D refers to the reference frame of the point placed on the back face of the door panel. The back face of the door is highlighted in light green and the dark green line represents the segment along which contact positions are sampled. 75

4.7 Illustration of the approach point (a), insertion point (b), and contact point (c). The insertion and approach points are generated by moving the gripper away from the door until its bounding sphere (shown in gray) no longer intersects the planes of the door's surfaces, represented by dashed gray lines. 76
4.8 A scene representing an estimated cabinet model colored in cyan, a ground truth cabinet model colored in black and a tool model colored in magenta. 77
4.9 Distribution of the number of actions required for a successful task across 1000 cabinets for each tested configuration in simulation. 83
4.10 Average number of actions per scene across 20 scenes used in the simulation experiments. 84
4.11 Distribution of actions per scene for the baseline configuration in the simulation experiments. 85
4.12 Number of required actions per scene for the real-world experiment with and without an offline calibration phase. 85
4.13 Euclidean distance between the origin of S_D on the corrected door panel and the ground truth in the real-world experiments. 86
4.14 A sequence of images showing the door-opening task in a real-world experiment. 87

List of Tables

2.1 Intersection over Union for each moving part . . . 23
2.2 State estimation accuracy for each moving part . . . 25

3.1 Cabinet properties used in simulation experiments . . . 43
3.2 Range of cabinet poses in real-world experiments . . . 45
3.3 Execution time analysis for different $n_{s,max}$ values ($N = 1000$ trials, $n = 40$) 51
3.4 Execution time analysis for different number of states n ($N = 1000$ trials, $n_{s,max} = 100$) . . . 51
3.5 Parameter values of the path planning algorithm used in the experiments 51
3.6 Results of the TSR, single-contact, multi-contact with SDF collision checking, multi-contact with FCL, and handle-to-multi-contact methods in the simulation experiments ($N = 1000$ trials) . . . 53
3.7 Dimensions of cabinets used in real-world experiments . . . 54
3.8 Summary of real-world experiment results ($N = 50$ trials per cabinet) . . . 54
3.9 Execution time comparison for different methods when a path is found ($N = 100$ trials) . . . 57
3.10 Execution times of the MC-SDF method using online vs. pre-computed contact pose graphs ($N = 20$ trials) . . . 58

4.1 Camera perturbation parameters and their associated noise variances . . . 78
4.2 Cabinet properties in real-world experiments . . . 79
4.3 User-defined parameters used in the sensitivity analysis and their tested values. The bolded value in each range was used as the baseline. . . . 80
4.4 Success metrics for each configuration across 1000 simulated cabinet-opening tasks . . . 81

1 Introduction

Robots are increasingly being deployed in demanding real-world environments, where they are expected to perform a wide range of tasks. While robots are still most commonly found in industrial settings, the adoption of service robots is steadily increasing in human-centered environments such as hospitality, transportation and logistics, agriculture, and many others (International Federation of Robotics, 2025). These environments are inherently complex and dynamic because they contain diverse objects whose poses and states are often uncertain and change over time. To operate autonomously and provide a satisfactory user experience, a service robot must therefore execute tasks reliably while continuously adapting to its surroundings.

Among the wide range of objects encountered in these environments, doors and drawers are particularly common examples of articulated objects. Such objects consist of multiple rigid parts connected by one or more translational or rotational joints, enabling relative motion between the parts (Pejić et al., 2023). Doors and drawers are ubiquitous in both household and industrial settings, and the ability to interact with them is often a prerequisite for higher-level tasks such as storing and retrieving items, accessing storage compartments, or entering new rooms. At the same time, manipulating doors and drawers remains challenging because it requires both accurate perception and reliable physical interaction, and small errors in the estimated pose or articulation state can lead to collisions or task failure.

For a robot to interact with an object such as a door or a drawer, it must robustly estimate the object's location and configuration within the environment in which it operates. This involves not only detecting the object, but also inferring its kinematic structure and current state (e.g., whether it is open and by how much). Human demonstrations provide a natural source of information for this process. By observing how people interact with articulated objects, a robot can extract useful information about the parts that move and how they move. In this thesis, the perception problem is addressed through a method for door and drawer detection and state estimation from image sequences of humans interacting with the objects. By analyzing the object's motion over time, the method constructs a model of an articulated object and estimates its state, providing the information required for subsequent manipulation.

In the existing research on the robotic door manipulation, it is often assumed that a handle providing a stable interaction point is present. However, handleless doors are increasingly common in modern furniture, yet there is limited research on this setting. In the absence of features such as handles, the robot must open the door by leveraging the surface of the door panel. Furthermore, service robots are often required to manipulate furniture in confined, cluttered environments, for example when operating in kitchens or inside narrow corridors. In such settings, approach directions can be limited and maintaining a single contact location on the door can be infeasible due to collisions and kinematic constraints. Therefore, this thesis addresses these issues by formulating the task of opening a cabinet door as a multi-contact path planning problem. By generating and searching over multiple end-effector configurations distributed over the door surface, the proposed approach removes reliance on a single precise contact and improves the likelihood of a successful execution.

Even with a robust door-opening strategy, failures can still occur in real-world settings because the planned motion relies on a reconstructed cabinet model whose accuracy is limited by several factors. Model errors may result from a combination of camera measurement noise, imperfect camera calibration, and perception and segmentation errors. These inaccuracies can lead to missed contacts when approaching the cabinet door, slipping during the opening motion, or unintended collisions with the cabinet. To address these issues, this thesis integrates visual, force, and tactile sensing in a framework that detects failures, recovers from them, and computes corrections to the camera parameters during execution. By refining the camera parameters, the system adjusts the reconstructed cabinet model, enabling more reliable opening trajectories and thereby reducing the likelihood of subsequent failures.

Supplementary material and pages corresponding to the main components of this thesis are available on the project website.[1]

1.1 Contributions

This thesis aims to develop new methods for robot interaction with articulated objects. In particular, the following original contributions to the scientific literature are proposed:

- **Method for detecting doors and drawers based on a sequence of RGB-D images of human demonstrations;**
- **Robot arm path planning method for multi-contact door opening;**
- **Method for opening doors with a robot arm based on RGB-D camera calibration using force-torque and tactile sensor data.**

1.2 Organization of the Thesis

This thesis is organized into five chapters. Chapter 1 is this introduction, which introduces the problem of robust interaction with articulated furniture in human-centered environments and motivates the challenges addressed in this thesis.

Chapter 2 presents a method for door and drawer detection and state estimation from RGB-D image sequences of human demonstrations. After reviewing related work, the chapter describes how motion information is integrated over time to construct an articulated object model and infer the object state, followed by experimental evaluation and discussion.

Chapter 3 focuses on robust handleless door opening and formulates the task as a multi-contact path planning problem. The chapter reviews relevant motion planning and door manipulation literature, introduces the proposed planning approach, and evaluates it in simulation and real-world experiments.

Chapter 4 extends the door-opening problem toward reliable real-world execution by integrating visual, force, and tactile sensing for failure detection and recovery. The chapter presents a door-opening framework with correction mechanisms that update camera parameters based on failed attempts and demonstrates the resulting improvement in real-world experiments.

Finally, Chapter 5 summarizes the main contributions and experimental findings and outlines directions for future work.

[1] `https://multi-contact-door.github.io`

2 Detection of Doors and Drawers from Human Demonstration

The ability to detect, localize and interpret the state of doors and drawers is a fundamental problem in robotic perception. While humans rely on instinctive spatial reasoning and intuition to recognize and manipulate articulated objects, robots depend on perception systems to interpret their environment. A large amount of existing research has focused on vision-based detection methods using RGB or RGB-D data, with the utilization of geometric reasoning, and more recently, deep learning. However, these methods often require large annotated datasets, known prior models of the objects or carefully designed object features for successful door and drawer perception.

In the absence of prior information about an object's location or structure, the robot requires guidance to initiate interaction. One effective way to provide this is through *Learning from Demonstration* (LfD), where a human user teaches the robot by showing how the task is performed. Among the various forms of LfD, a particularly effective approach for acquiring object information involves a human demonstrating the manipulation task while the robot observes through its camera. In the context of opening doors and drawers, this allows the robot to infer the kinematic properties of the object and its location within the room.

To address this challenge, this chapter introduces a novel method that allows a robot to efficiently determine the location of doors and drawers within a given environment map and assess their current state. The approach utilizes human demonstrations to teach the robot the location of doors and drawers (as shown in Figure 2.1) within the environment, allowing the method to generate corresponding models and incorporate them into the map. During the teaching process, a human demonstrates how to open each door and drawer, and the proposed algorithm analyzes the resulting image sequences to create corresponding door and drawer models, which are then positioned within the environment map. Once the environment map has been augmented with the models, the robot can later observe the scene and reliably infer whether a door or a drawer is open or closed, and to what extent. This method is referred to as the *Door and Drawer Detector* (DDD), which is composed of two complementary algorithms. The first, DDD-THD (*Teaching by Human Demonstration*), processes demonstration sequences to detect the moving part and generate its model. The second, DDD-SE (*State Estimator*), operates at run time, using a single RGB-D image to estimate the current state of a given door or drawer. Together, these two components provide the foundation for recognizing and tracking articulated furniture, a capability that will later support motion planning and interaction tasks.

Unlike the approach presented in Rühr et al., 2012, which also creates a door or a drawer model from a 3D image sequence, the proposed approach does not rely on information about the robot gripper during the robot action. Instead, this approach uses a generic model of the door panel or drawer front and fits the point clouds captured by a 3D camera to this model. The proposed method is based on generating candidate models by hierarchical evidence accumulation over a depth image sequence and evaluating these candidate models by projecting them onto the image and calculating the hypothesis evaluation score, similar to that used in Cupec et al., 2020 and Aldoma et al., 2016. No prior work has been identified that utilizes this approach to solve the problem under consideration. The created model is used to estimate the state of the door or drawer based on a single depth image.

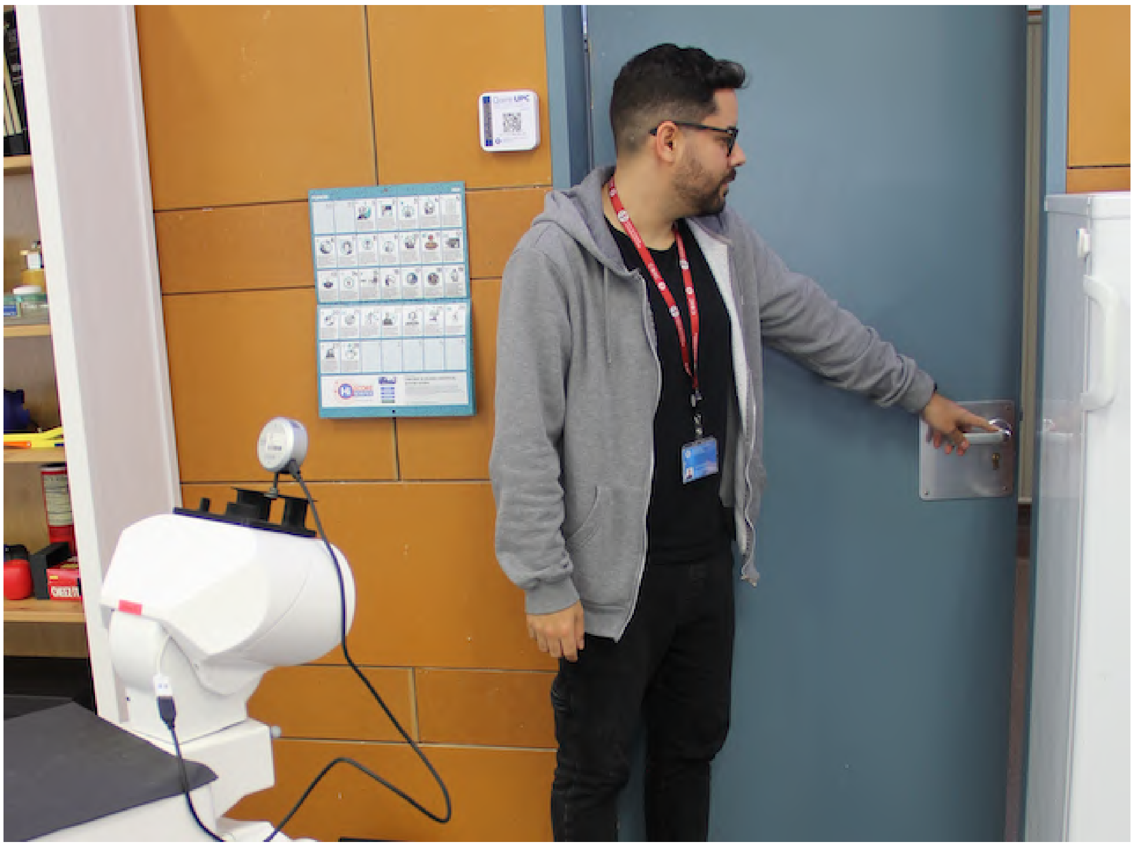

Figure 2.1: A human demonstrates to a robot the movement of a door. The robot learns where the door is located and its direction of opening.

2.1 Related Work

Research on articulated objects has produced a variety of approaches for detecting and segmenting movable parts, including doors and drawers. This section reviews the existing relevant literature, beginning with general methods for articulated object detection and kinematic parameter estimation, and then focusing on approaches specifically designed for door and drawer detection. It also briefly touches on Learning from Demonstration strategies, as these approaches can support effective interaction with doors and drawers.

2.1.1 Articulated Object Detection

When it comes to the detection of articulated objects, throughout the years, many approaches have been utilized. Previous research on the detection and manipulation of articulated objects has been extensively discussed in Pejić et al., 2023, which compares image-processing-based methods and benchmark datasets for detecting and segmenting articulated objects, determining their joint parameters, and manipulating them with a robot.

Early approaches often relied on geometric cues and simplified object shapes to analyze motion. In Gelfand and Guibas, 2004, motion slippage within a single 3D point cloud was used to identify the types of joints present in an object. A related method was later introduced in Xu et al., 2009, extending the idea to more complex mesh models. However, both approaches were limited to synthetic data without considering real-world scenes captured by cameras. These ideas were further extended in Mitra et al., 2010 and Sharf et al., 2014, where the authors performed shape analysis by decomposing complex 3D CAD objects into parts and detecting their functional mobilities. In Yuan et al., 2016, the authors demonstrated that sequences of point clouds of moving articulated objects can be leveraged to identify their parts. Their method propagates local segmentations, obtained by clustering points in individual frames, to neighboring frames. These segments are then grouped spatially and temporally to obtain coherent motion parts. While effective for discovering object segments, this approach does not explicitly model the kinematic relationships between the parts.

Later works explored deep neural networks for object detection in general, which were then applied to detection of articulated objects. Qi et al., 2017b; Qi et al., 2017a introduced PointNet and PointNet++, enabling end-to-end learning directly from unordered point clouds. Graham, Engelcke and Maaten, 2018 proposed Submanifold Sparse Convolutional Networks (SSCNs), which enable efficient convolution operations directly on sparse 3D data. This approach allows large-scale architectures such as Sparse U-Nets to be trained on point clouds, bypassing the need for handcrafted features and supporting discriminative part segmentation in articulated objects. However, these methods do not capture the kinematic models and they require large amounts of data to train.

More recent approaches directly target the motion of parts. Huang et al., 2021 proposed a multi-view approach that uses scene flow to establish correspondences between two point clouds, after which it infers part segmentations and their rigid transformations. Similarly, Yi et al., 2018 introduced a method that takes a pair of point clouds of the same object in different configurations, predicts point correspondences and deformation flows between them, and iteratively refines part segmentations by grouping points that move together. On the other hand, Kawana, Mukuta and Harada, 2021 utilized an unsupervised model that segments parts and predicts their poses from a single point cloud, learning implicit fields that represent both shape and motion. However, the method was trained and tested only on synthetic data. Gadre, Ehsani and Song, 2021 demonstrated that articulated object parts can be discovered by reasoning over differences between multiple object configurations observed during interaction. This allows motion masks of moving parts to be inferred using only RGB data. However, the approach depends on learned action policies trained in simulation and requires multiple repetitions of exploratory motions.

2.1.2 Estimation of Kinematic Parameters

Once parts are detected, the next step is to estimate the kinematic parameters of an articulated object, which correspond to the direction of translation or axis of rotation of a moving part with respect to the rest of the object.

Li et al., 2016 proposed mobility fitting using RANSAC, where candidate joint models, such as prismatic or revolute joints, are fit to a sequence of point clouds of moving objects. However, this method may fail when motion is small or ambiguous. In Zeng et al., 2021, a method is proposed that predicts motion fields and connectivity between object parts, allowing joint axes and motion directions to be inferred directly from RGB-D observations. While the method links segmentation and motion estimation, it requires consistent ground truth annotations, which are difficult to obtain in real-world scenarios.

Liu et al., 2020 proposed GC-Pose, an approach that segments moving parts and estimates their relative motions from a single-view video sequence. These motion estimates are then accumulated and refined across frames to recover the configuration of an articulated object. However, the method is designed primarily for revolute joints and does not address prismatic joints.

Jain et al., 2021; Zhang et al., 2021 proposed neural networks that estimate joint parameters from series of RGB-D images or point clouds, where a motion of an object is captured. However, these methods typically assume that the articulated object is separated from the background and do not work with complex scenes without additional segmentation or detection methods.

Wang et al., 2019 proposed a framework for learning articulated structures by projecting multi-modal features from RGB-D data into a shared embedding space. This representation encodes both geometric and motion information, which makes it possible to recognize similar types of movements across different object categories. While this method focuses on a general embedding, Li et al., 2020 extends this work to localize objects and their parts from 3D point clouds, predicting each part's pose and kinematic state. The method learns features that capture both local geometry and motion cues, allowing it to infer how different parts are arranged and how they can move. However, the method is sensitive to occlusions and assumes that all instances within a given object category share the same structure, including the number of parts and the joint types.

2.1.3 Door and Drawer Detection

Detection of doors and drawers shares many principles with the general detection of articulated objects and the estimation of kinematic parameters but focuses mostly on functional structures in household environments.

Chen and Birchfield, 2008 and Kwak, Arisumi and Yokoi, 2011 used RGB images and simple geometric properties to identify rectangular regions that look like doors. Sturm et al., 2010 used stereo vision for this task, segmenting surfaces in point clouds and aligning planar hypotheses with the data to detect doors and drawers. Similarly, Quintana et al., 2018 combined color segmentation with geometric properties of the objects on point clouds to identify cabinet doors. Other geometry-based methods have explored plane extraction and shape analysis in RGB-D data (Meyer Zu Borgsen et al., 2014; Habib, 2021; Vlaminck et al., 2016; Skulimowski, Owczarek and Strumillo, 2017; Banerjee et al., 2015). In cluttered or textured environments, these methods often suffer from limited robustness and are usually limited to standardized furniture models.

In contrast, Rühr et al., 2012 proposed an approach that uses visual information collected during robot manipulation with doors and drawers to estimate their kinematic model. However, the approach relies on the presence of handles for manipulation and is constrained by specific sensors of the robot platform.

With the rise of convolutional neural networks, more robust detection frameworks were developed. Llopart, Ravn and Andersen, 2017 introduced a YOLO-based architecture to detect door handles from RGB-D images, segmenting regions of interest on doors and drawers. Arduengo, Torras and Sentis, 2021 proposed a method that utilized YOLOv3 models trained on a custom dataset called DoorDetect to detect doors with handles. Similarly, Kim et al., 2022 applied YOLOv3 for mobile robot navigation, detecting doors in indoor environments. Ramôa, Alexandre and Mogo, 2020; Ramôa et al., 2021 proposed methods based on PointNet for real-time door detection and state estimation on low-power hardware such as the NVIDIA Jetson Nano.

To support such methods, several datasets for door and drawer perception have been introduced. The DoorDetect dataset proposed in Arduengo, Torras and Sentis, 2021 provides annotated RGB images of doors and handles, while the HoDoor dataset proposed in Šimundić et al., 2023 provides labeled RGB-D data with classification of door-opening types.

Despite significant progress, current methods can often struggle with generalization to different door and drawer designs, sensitivity to occlusions and clutter, or need large-scale diverse datasets to operate reliably. These limitations motivate the need for more methods that can operate robustly in realistic household environments.

2.1.4 Learning from Demonstration

Another line of research relevant to doors and drawers is Learning from Demonstration (LfD). In LfD, a robot learns to replicate human actions by observing or being guided through examples, enabling it to acquire diverse strategies for manipulation.

Early approaches often relied on kinesthetic teaching, where a human physically guided the robot to perform a task. Calinon et al., 2010; Racca et al., 2016 demonstrated how trajectories for manipulation tasks could be encoded into probabilistic models from kinesthetic demonstrations. These methods enabled robots to reproduce smooth motions but required direct physical access to the robot.

More advanced approaches utilize observations based on vision systems. Welschehold, Dornhege and Burgard, 2017 presented a system where a mobile robot observed humans opening doors with a 3D camera and subsequently replicated the action. Similarly, Sermanet, Xu and Levine, 2017 demonstrated that video demonstrations could be used with deep reinforcement learning for manipulation tasks. In Li, Baum and Brock, 2023, a method was proposed where a robot learned manipulation policies from a single demonstration, relying on augmentation of demonstration data to generalize across variations. Although only one demonstration is required, the augmentation process depends on prior knowledge of the object's constraints and can only add a limited amount of task information.

2.2 Augmented Environment Map

To perform meaningful interaction with its surroundings, a service robot must localize itself within a map of the environment, but also understand the elements that it encounters in the environment. While standard environment maps used for robot navigation typically include obstacles such as walls and furniture contours, they do not contain semantic or kinematic information about the objects like doors and drawers. To bridge this gap, this chapter introduces an *augmented environment map*, which enhances a map of the robot's surroundings by embedding articulated object models with their geometric and kinematic properties. An example of the augmented environment map is shown in Figure 2.2.

The method proposed in this chapter detects articulated objects such as doors and drawers and inserts corresponding detected models into the environment map. These *door and drawer models* (DMs) provide the robot with information about the dimensions of the detected object and the position and orientation of the joint axis, expressed in the global environment frame S_W.

2.2.1 Door Model

The *door model* (DM) used in this chapter consists of a door panel, represented as the movable part connected to the stationary part of furniture by a revolute joint. The reference frame S_A is centered on the joint axis, with its z-axis aligned with the joint axis.

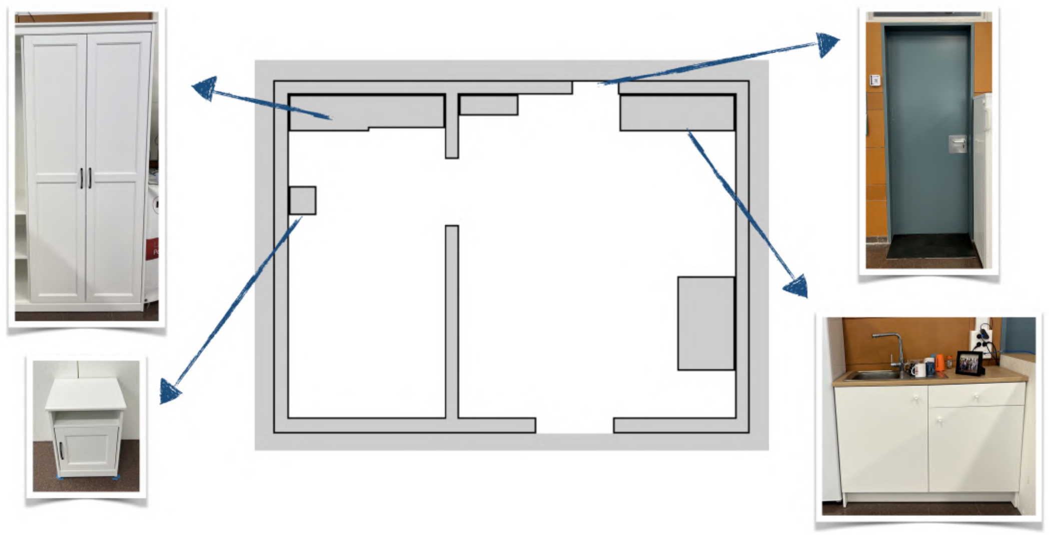

Figure 2.2: An example of an augmented environment map showing the locations of different pieces of furniture in the Institut de Robòtica i Informàtica Industrial (IRI), Barcelona.

The door panel itself is modeled as a cuboid that rotates about the joint axis, as illustrated in Figure 2.3. A second reference frame S_B is placed at the center of the cuboid, with its axes parallel to the cuboid edges. The cuboid's dimensions are described by the vector $l = \begin{bmatrix} l_x & l_y & l_z \end{bmatrix}^\top$, whose elements correspond to its size along the x-, y-, and z-axes, respectively. The origin of S_A is defined as the orthogonal projection of the origin of S_B onto the joint axis. The door state is expressed by the angle φ, defined as the angle between the x-axes of S_A and S_B. When the door is closed, the x-axes of S_A and S_B are parallel, which defines the zero-state ($\varphi = 0$), as illustrated in Figure 2.3. The relative position of the joint axis with respect to the panel is encoded by the vector $r = \begin{bmatrix} r_x & r_y \end{bmatrix}^\top$, representing the x- and y-coordinates of the panel center in the zero-state with respect to S_A, as shown in Figure 2.3. In the zero-state, the pose of S_B with respect to S_A is represented by the following homogeneous transformation matrix (HTM):

$$ {}^{A}T_{B}\left(0\right) = \begin{bmatrix} I_3 & {}^{A}t_B \\ 0 & 1 \end{bmatrix}, \tag{2.1} $$

where I_3 is the 3×3 identity matrix and ${}^{A}t_B = \begin{bmatrix} r_x & r_y & 0 \end{bmatrix}^\top$. When the door is rotated by an angle φ, the transformation becomes:

$$ {}^{A}T_{B}\left(\varphi\right) = T_z\left(\varphi\right) {}^{A}T_{B}\left(0\right), \tag{2.2} $$

where $T_z(\varphi)$ represents the rotation around the z-axis by angle φ. Finally, the position and orientation of S_A with respect to the world frame S_W are given by an HTM ${}^{W}T_A$.

2.2.2 Drawer Model

The drawer front is modeled as a cuboid with an associated reference frame S_B and size vector l, defined in the same way as for the door panel model. The remaining parts of the drawer are not explicitly represented. The drawer extends along the x-axis of S_B, with its state defined by the degree of extension ω, where $\omega = 0$ corresponds to a fully closed drawer. In the zero-state, the model reference frame S_A is aligned with S_B. For a

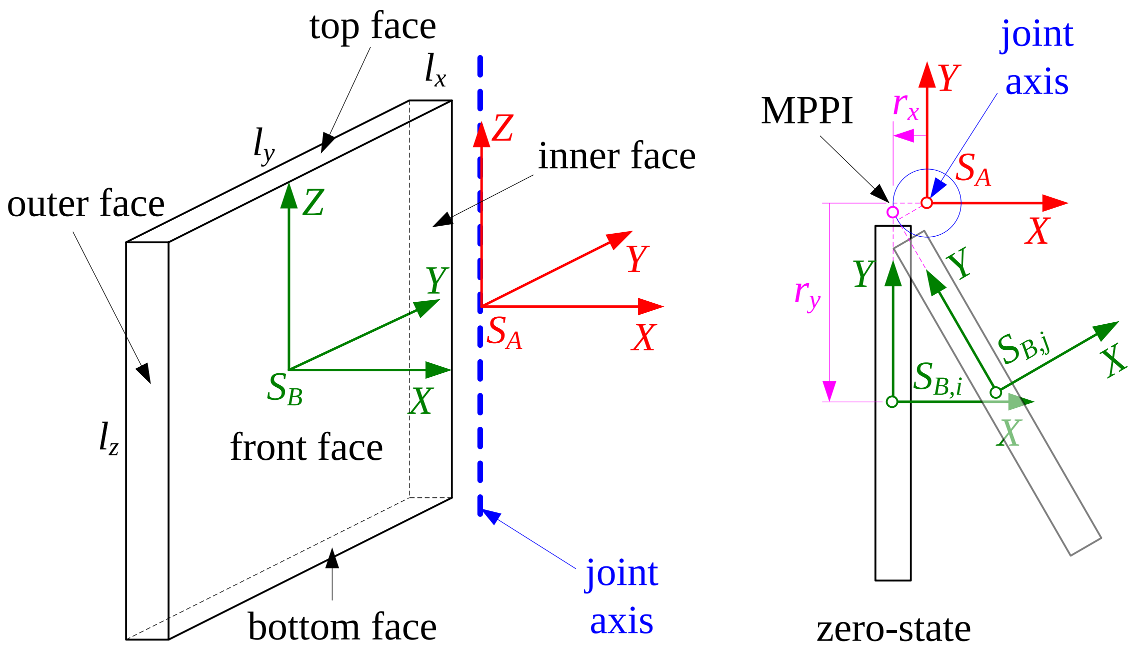

Figure 2.3: Door model

given extension ω, the pose of S_B with respect to S_A is:

$$
{}^{A}T_B(\omega) = \begin{bmatrix} I_3 & {}^{A}t_B\,(\omega) \\ 0 & 1 \end{bmatrix}, \tag{2.3}
$$

where ${}^{A}t_B\,(\omega) = \begin{bmatrix}\omega & 0 & 0\end{bmatrix}^\top$ represents a translation along the x-axis of S_A. An illustration of the drawer model is provided in Figure 2.4.

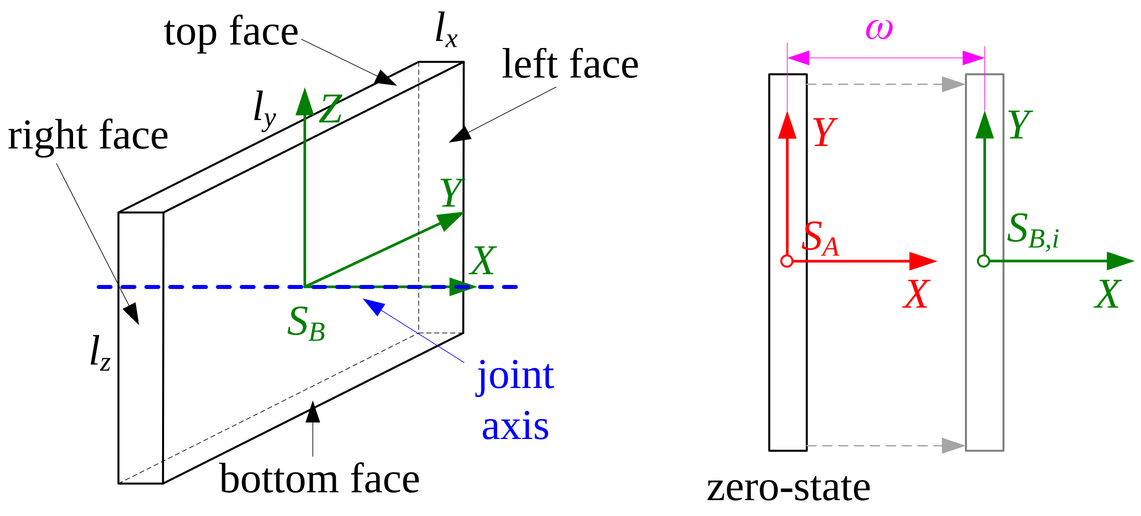

Figure 2.4: Drawer model

2.3 Teaching by Human Demonstration

This section presents the DDD-THD algorithm. Its input consists of two elements: (i) the current pose of the robot with respect to the environment map reference frame, provided by a robot localization method, and (ii) a sequence of RGB-D images showing a human opening a door or a drawer. The sequence must be recorded with a stationary camera to ensure that the only moving elements in the scene are the human demonstrator and the moving part of the articulated object. The algorithm first detects and segments the human in each image using TensorMask (Chen et al., 2019) and then removes the human pixels so that the only moving object in the scene is the moving part. From the resulting images, it generates one or more *moving part hypotheses* (MPHs) per frame and integrates these hypotheses across the sequence to form *door/drawer hypotheses* (DHs). A confidence score is assigned to each hypothesis, and the highest-scoring one is used to construct a *door/drawer model* (DM), which is then inserted into the environment map. The algorithm consists of the following steps applied to a sequence of RGB-D images:

1: Removal of people from the images
2: Generation of MPHs for images in the sequence
3: Integration of MPHs into DHs
4: Assigning a confidence value to the DHs
5: Selection of the DH with the highest confidence value
6: Creating a DM and inserting it into the environment map

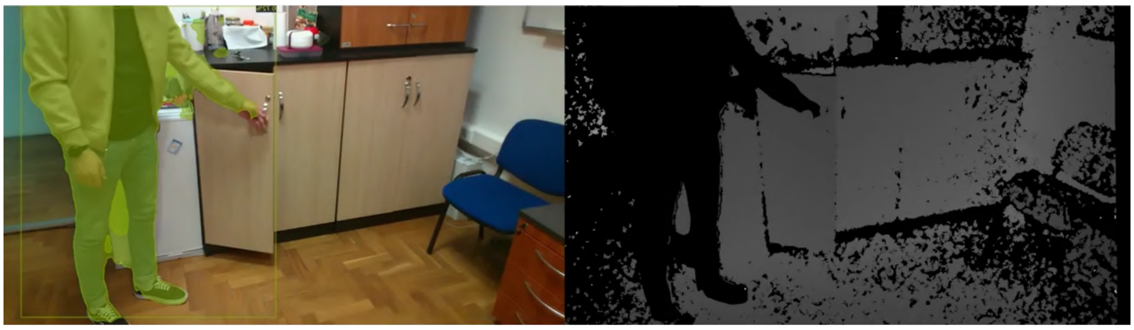
Figure 2.5: Example of human removal from depth images using TensorMask. The left image shows the segmented human, while the right image displays the depth map with the corresponding pixels removed.

2.3.1 Clustering

The clustering procedure used throughout this chapter is based on a *complete-linkage* agglomerative clustering approach with a fixed distance threshold. The goal of this clustering method is to group data points into clusters such that no pair of points within the same cluster is farther apart than a given threshold. It should be noted that this clustering procedure differs slightly from the standard complete-linkage algorithm in the order in which merges are considered. This difference is described below.

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of n data points in $\mathbb{R}^d$. The algorithm begins by computing a symmetric distance matrix $E \in \mathbb{R}^{n \times n}$, where $E(i,j)$ represents the pairwise distance between points x_i and x_j. The particular distance matrices will be explained in more detail in the following sections.

Initially, each data point is placed in its own cluster. The algorithm then considers all unordered pairs of points (x_i, x_j) with $i < j$, which are sorted in ascending order of their distance $E(i,j)$. It should be noted that the algorithm does not merge based directly on point pairs but instead uses these point pairs to decide which *clusters* should be merged. For a given pair (x_i, x_j) in the sorted list, let $\mathcal{C}_a$ and $\mathcal{C}_b$ be the current clusters containing x_i and x_j, respectively. If x_i and x_j already belong to the same cluster, the pair is skipped since no further merge is necessary. Otherwise, $\mathcal{C}_a$ and $\mathcal{C}_b$ are considered as candidates for merging. The merge is performed only if the maximum pairwise distance between any point in $\mathcal{C}_a$ and any point in $\mathcal{C}_b$ is within the threshold τ. If the condition is satisfied, $\mathcal{C}_b$ is merged into $\mathcal{C}_a$, and the cluster labels are updated accordingly. The algorithm stops evaluating further point pairs (x_i, x_j) once the sorted list reaches distances greater than the threshold τ.

The main difference from standard complete-linkage clustering lies in how candidate merges are selected. In standard complete-linkage clustering, the distance between two clusters is defined as the maximum distance between any pair of points across the two clusters, and at each step the algorithm merges the pair of clusters with the smallest such maximum distance. In contrast, the procedure described here uses the minimum point-pair distance between the pair of clusters to propose candidate merges. The complete-linkage criterion is then enforced by merging the clusters only if their maximum inter-point distance is within the threshold τ.

The clustering process is described in Algorithm 1.

Algorithm 1 Complete-linkage clustering with distance threshold

```
 1: procedure CLUSTERING($\mathcal{X}, \tau$)
 2:     $n \leftarrow |\mathcal{X}|$
 3:     $E \leftarrow$ pairwise distances between all points in $\mathcal{X}$
 4:     $\mathcal{P} \leftarrow \{(i,j) \mid 0 \le i < j < n\}$
 5:     Sort $\mathcal{P}$ by $E(i,j)$ in ascending order
 6:     Initialize clusters: $\mathcal{C}_k \leftarrow \{k\}$ for $k = 0$ to $n-1$
 7:     Initialize labels: $L_k \leftarrow k$ for $k = 0$ to $n-1$
 8:     repeat
 9:         $(i,j) \leftarrow$ first element of $\mathcal{P}$
10:         remove $(i,j)$ from $\mathcal{P}$
11:         if $E(i,j) > \tau$ then
12:             break
13:         end if
14:         if $L_i \neq L_j$ then
15:             $A \leftarrow \mathcal{C}_{L_i}, B \leftarrow \mathcal{C}_{L_j}$
16:             if $\max_{p \in A, q \in B} E(p,q) \le \tau$ then
17:                 $\mathcal{C}_{L_i} \leftarrow A \cup B$
18:                 $\mathcal{C}_{L_j} \leftarrow \emptyset$
19:                 for all $q \in B$ do
20:                     $L_q \leftarrow L_i$
21:                 end for
22:             end if
23:         end if
24:     end repeat
25:     return $\{\mathcal{C}_k \mid \mathcal{C}_k \neq \emptyset\}$
26: end procedure
```

2.3.2 Moving Part Hypotheses

In a given image from the RGB-D sequence, moving regions are detected by comparing that image with other images in the sequence within a predefined temporal window. This temporal window corresponds to the number of images adjacent to the current image. Two compared RGB-D images are represented by 3D point clouds, whose points are used to determine the most prominent direction of motion of the object. To improve computational efficiency, both RGB-D images are first subsampled. This subsampling is performed by overlaying a grid with cells of size $n_c \times n_c$ over the depth image, where n_c is the cell size in pixels. Within each cell, representative points are selected based on the consistency of surface normals. If all points within a cell have similarly oriented normals, a single representative point is chosen (see Figure 2.6.a)). However, if the cell contains surfaces with different orientations, such as at an edge or a corner, multiple representatives may be retained (see Figure 2.6.b)). This strategy ensures that important geometric transitions are preserved while reducing the number of points used for further processing.

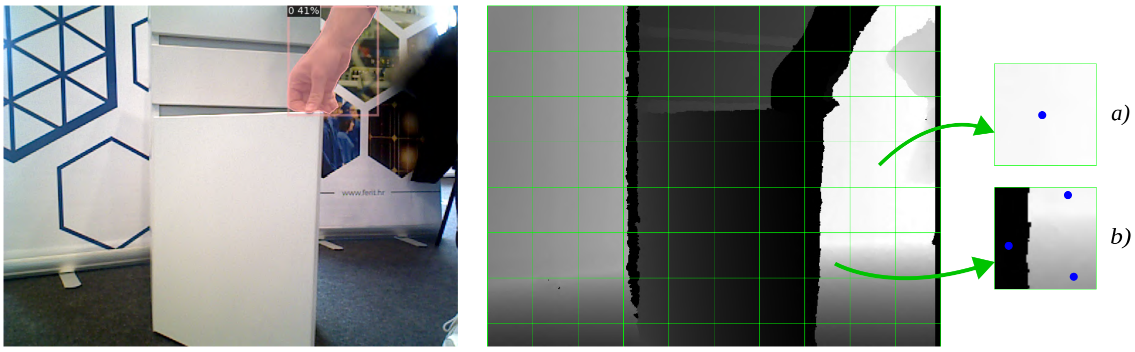

Figure 2.6: Illustration of subsampling of an RGB-D image. The depth image (right) is overlaid with a grid, where representative points are selected within each cell based on surface normal consistency. Cell (a) contains a single planar surface and is represented by one point, while cell (b) includes multiple surfaces at an edge and is represented by multiple points.

Next, for each pair of images, a set of reliable point associations is established using a nearest-neighbor strategy, where each point in the first point cloud is matched to its closest point in the second point cloud based on Euclidean distance. These correspondences are then used to compute displacement vectors, each representing the difference between a point in the first point cloud and its matched point in the second. To reduce the impact of the camera noise in the depth images, only the displacement vectors with length above a user-defined threshold are considered. These displacement vectors then pass through a voting procedure, in which the displacement directions are put into a uniform grid of cells. Each vector casts a vote for the corresponding cell and a cell with the highest number of votes is selected as the one containing the dominant displacement direction. The vector associated with the center of this cell represents the *dominant displacement vector*, illustrated with a blue line in Figure 2.7. Subsequently, the algorithm filters the computed displacement vectors. Only those vectors that are sufficiently long and are aligned with the dominant displacement vector are retained. The endpoints of these valid vectors are referred to as *displacement points*, which are marked green in Figure 2.7.

For each point cloud in the predefined temporal window, candidates for moving part hypotheses are computed based on the segmentation of the surfaces, together with the motion information explained earlier. The surfaces are segmented into approximately planar patches using a suitable segmentation method. For the detection of planar patches, the method proposed in Cupec, Filko and Nyarko, 2017 is used, as shown in Figure 2.9. Planar patches containing a sufficient number of displacement points are considered as candidates for the front of a moving part.

For each selected planar surface, a *moving part hypothesis* (MPH) is generated by leveraging outputs from the method proposed in Cupec, Filko and Nyarko, 2017. In addition to detecting planar surfaces, the method estimates a surface normal for each detected planar patch. It further extracts contours of image points at depth discontinuities and

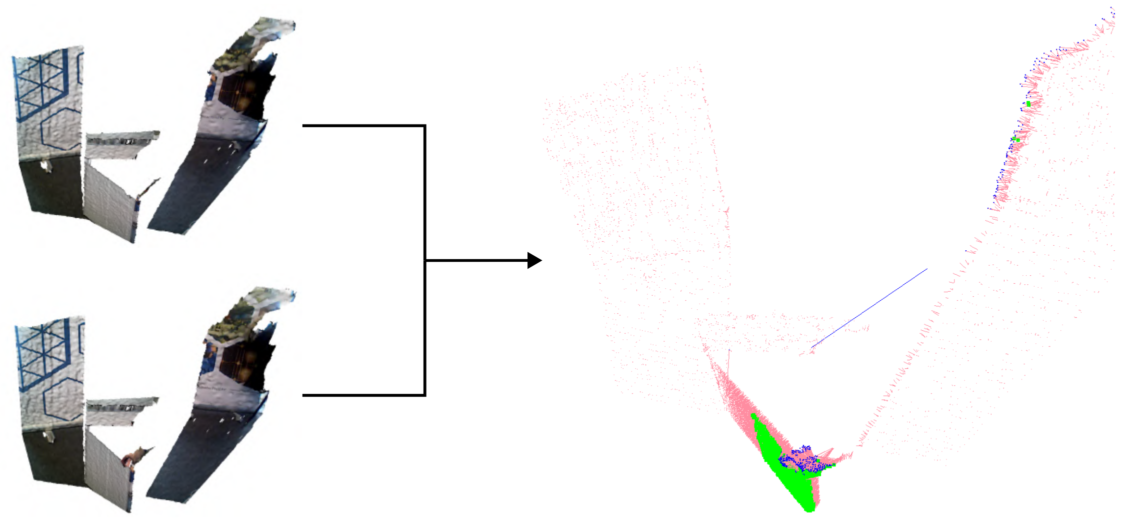
Figure 2.7: Computation of the dominant displacement vector from two point clouds within a predefined temporal window. Displacement vectors obtained via nearest-neighbor matching are shown in pink, with the dominant displacement vector highlighted in blue. Vectors that are sufficiently long and are aligned with the dominant direction are marked with green endpoints, while misaligned vectors are marked with blue endpoints.

approximates these contours by line segments. An example is given in Figure 2.8, where a planar surface representing the front face of a moving region is outlined in black, and the line segments representing the boundaries of the planar surface are depicted in red, green, and blue. These line segments are used to determine the orientation of the cuboid associated with the MPH, as well as its dimensions l_y and l_z. The element l_x represents the thickness of a door panel or a drawer front. This element is set to a constant value of 0.018 m, which fits the majority of real cases within a measurement noise range.

For each line segment, a bounding box is computed such that its edges align with the orientation of the line segment while enclosing the corresponding planar surface with minimal area. The normal vector of the planar surface aligns with the x-axis of this bounding box. To determine the y-axis, a plane is formed by the line segment and the camera center. The cross product of the plane's normal with the x-axis defines the y-axis. Then, the z-axis is obtained as the cross product of the x- and y-axes. Among all candidate bounding boxes generated for the observed surface, the one with the smallest area in the yz-plane is selected to define S_B, as illustrated by the blue bounding box in Figure 2.8.

The result is a generated MPH represented by a tuple $h = ({}^{C}T_B, l, k)$, where ${}^{C}T_B$ represents the pose of S_B with respect to the camera frame S_C, l is the size vector defined in Section 2.2, and k is the index of the image in the input image sequence from which the MPH is generated. An example of an MPH is shown in Figure 2.9.

2.3.3 Door/Drawer Hypotheses

Door/Drawer hypotheses (DHs) are generated by integrating MPHs over the image sequence. A DH is represented by a tuple $H = (\eta, {}^{C}T_A, l, r, \mu, \Theta)$, where η is the object type: *door* or *drawer*, ${}^{C}T_A$ represents the pose of S_A with respect to S_C, defined in Section 2.2, l and r are vectors also defined in Section 2.2, $\mu \in \{-1, 1\}$ is the *axis-side parameter* indicating whether the joint axis lies to the left or right of the door panel, and Θ is the *state sequence*. For a given door, it is assumed that all non-zero states have only

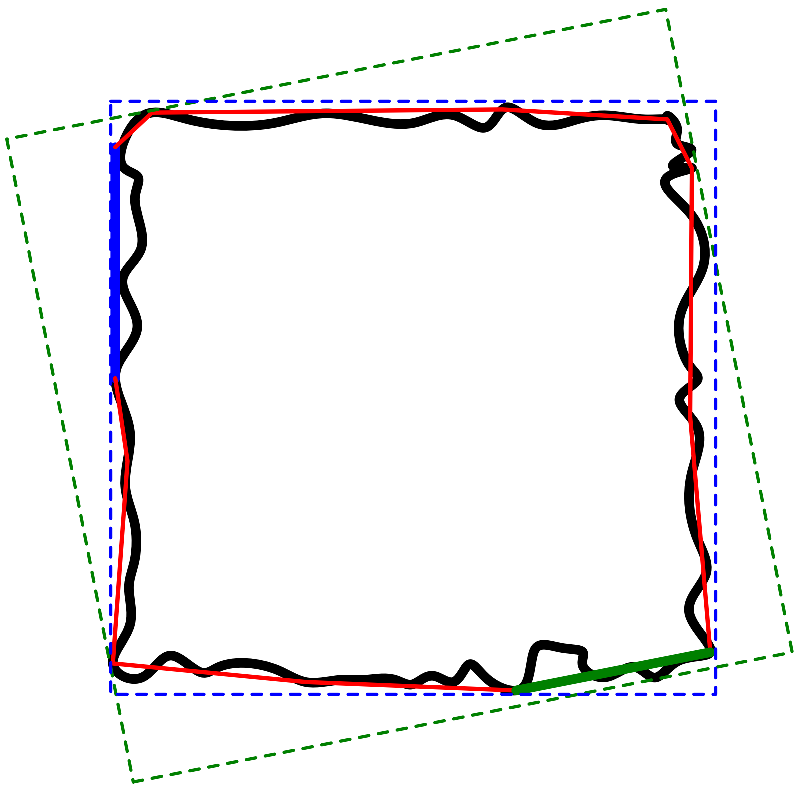

Figure 2.8: The front face of an MPH whose boundary is approximated by line segments and two bounding boxes (dashed lines). The blue bounding box is aligned with the blue line segment and the green bounding box is aligned with the green line segment.

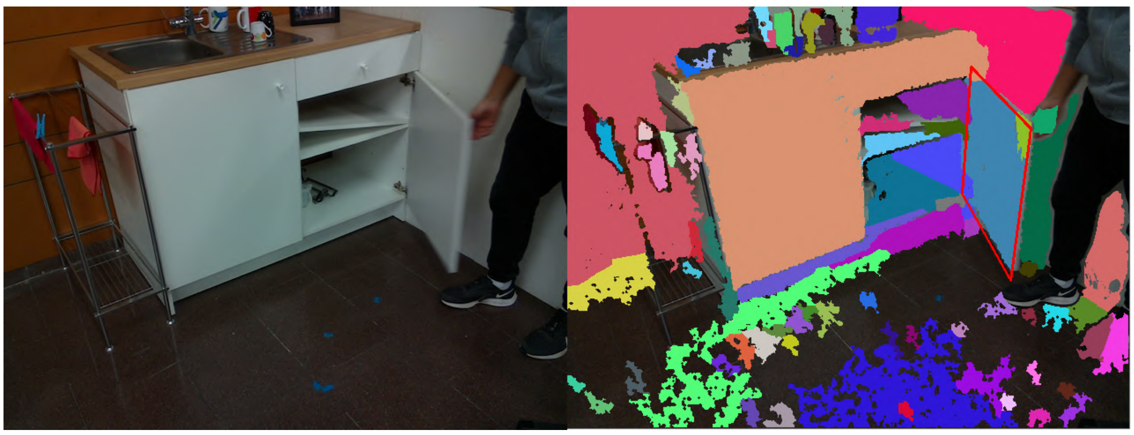

Figure 2.9: Input image (left) and a moving part hypothesis denoted by red lines (right). Colored regions in the right image represent detected planar patches.

positive or only negative values. The axis-side parameter μ determines this sign convention for the feasible door states. In the case of a drawer hypothesis, μ is always 1 and $r = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top$. Each state in the sequence Θ is represented by a pair, either (k, φ) for doors or (k, ω) for drawers. Here, k denotes the index of the image in which the moving part appears, while φ represents the door's angular state and ω represents the drawer's extension state.

Door Hypotheses

The integration of MPHs into door hypotheses is done by a hierarchical clustering procedure. First, z-axis hypotheses are generated by forming pairs of MPHs and computing a z-axis candidate for each such pair. Let (h_i, h_j) be a pair of MPHs. Each of these two MPHs is associated with an RF S_B, as explained in Section 2.3.2. Since the z-axis of S_A is supposed to be perpendicular to the x-axis of S_B in each door state, a z-axis candidate is computed as the unit vector perpendicular to the plane spanned by the x-axes of the RFs $S_{B,i}$ and $S_{B,j}$ associated with the MPHs h_i and h_j. In this way, a set of z-axis candidates H^z is formed. Let $z_i, z_j \in H^z$ denote a pair of z-axis candidates from the set H^z. A symmetric distance matrix $E_z\,(i,j)$ is constructed where each element is the angular distance between z_i and z_j, expressed as:

$$E_z\,(i,j) = 1 - |z_i^\top z_j|. \tag{2.4}$$

This metric is 0 when the vectors are collinear. The resulting matrix serves as an input to the clustering procedure described in Section 2.3.1, which groups together z-axis candidates that are similar within a predefined angular threshold. Thus, each resulting cluster $\mathcal{C}^z$ provides a set of candidate directions that are consistent with one another. To obtain a single representative z-axis orientation, all MPHs that were used to form the cluster $\mathcal{C}^z$ are collected. Their x-axes are expected to lie approximately in a common plane, which is orthogonal to the door's rotation axis. To identify the normal to this plane, which represents the z-axis direction, a covariance matrix of the x-axes is constructed, and principal component analysis (PCA) is applied. The eigenvector associated with the smallest eigenvalue is then selected as the estimated z-axis, as it best captures the direction orthogonal to the observed x-axes. This step identifies the most likely axis direction of the door's rotation, but its position remains to be determined.

The next step defines the position of the joint axis. Let $\chi\,(\mathcal{C}^z)$ be the set of MPHs involved in the formation of a cluster $\mathcal{C}^z$. From each such set, one or more joint axis hypotheses are generated. Consider a pair of MPHs $(h_i, h_j) \in \chi\,(\mathcal{C}^z)$. A joint axis of a door is usually very close to the line representing the intersection of the yz-planes of RFs $S_{B,i}$ and $S_{B,j}$, as shown in Figures 2.3 and 2.10. The *moving part plane intersection* (MPPI) is defined as the point representing the intersection of this line with a plane Π, which is constructed to be perpendicular to the z-axis hypothesis and to pass through the origin of the camera reference frame S_C. By introducing Π, all intersection lines are consistently projected onto a common reference plane, which simplifies the comparison and clustering of candidate joint axis positions. Since this plane Π is shared across all MPPI calculations for a given $\mathcal{C}^z$, all MPPIs lie on it. The MPPI captures the likely x- and y-coordinates of the joint axis, while the z-axis orientation is provided by the cluster $\mathcal{C}^z$. In this step, the z-coordinate of the joint axis position is not yet known. An illustration of the MPPI is shown in Figure 2.10.

Since the MPPI is usually very close to the actual joint axis, the position of the joint axis can be estimated by clustering MPPIs computed from pairs of MPHs from the set $\chi\,(\mathcal{C}^z)$. To cluster MPPIs obtained from the MPH pairs, a pairwise distance matrix

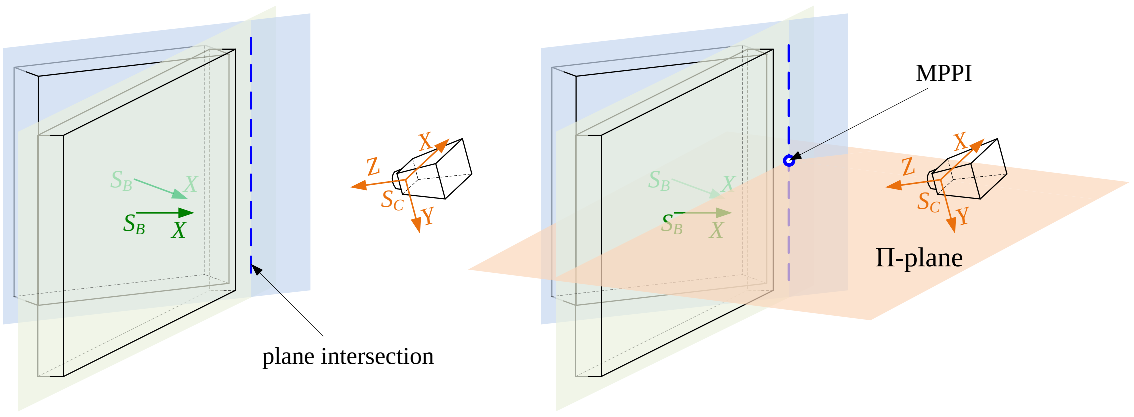

Figure 2.10: Illustration of the moving part plane intersection (MPPI). A plane intersection line (blue) is first computed from the intersection of two moving surfaces (light green and light blue). Then, a plane Π is defined such that it is perpendicular to the z-axis hypothesis and passes through the origin of the camera reference frame S_C. The MPPI is obtained as the intersection point between the line and the plane Π.

E_{MPPI} is constructed. Each element $E_{\text{MPPI}}(i, j)$ contains the squared Euclidean distance between two MPPIs, defined as:

$$E_{\text{MPPI}}(i, j) = \|p_i - p_j\|^2, \tag{2.5}$$

where p_i and p_j are the MPPIs obtained from two MPH pairs. Clustering MPPIs groups together points that are spatially consistent with a single joint axis. Therefore, each resulting cluster $\mathcal{C}^{\text{a}}$ defines a hypothesis for the axis position in the plane Π, with its representative being the centroid of the clustered MPPIs.

Let $\chi(\mathcal{C}^{\text{a}})$ be the set of MPHs involved in forming the cluster $\mathcal{C}^{\text{a}}$. It is assumed that for all MPHs $h_i \in \chi(\mathcal{C}^{\text{a}})$, the yz-plane of their respective RF $S_{B,i}$ lies at the same distance r_x from the joint axis. This (signed) distance can be computed by

$$\delta(h_i, p) = x_{B,i}^{\top}(c_{B,i} - p), \tag{2.6}$$

where $x_{B,i}$ and $c_{B,i}$ are the x-axis and the origin of $S_{B,i}$, both projected onto the plane Π, and p is a candidate point on the joint axis that intersects the plane Π. The distance δ expresses how far the yz-plane of a given MPH h_i is from a candidate joint axis location p. For each cluster $\mathcal{C}^{\text{a}}$, the position of the joint axis is estimated by calculating the point p and the distance r_x that minimize the cost function

$$E(p, r_x) = \sum_{h_i \in \chi(\mathcal{C}^{\text{a}})} \left(\delta(h_i, p) - r_x\right)^2. \tag{2.7}$$

The minimization of this least-squares cost encourages all RFs $S_{B,i}$ to align with a joint axis that is parallel to the z-axis hypothesis from $\mathcal{C}^{\text{z}}$ and passes through the optimal point p in the plane Π. This step finds the representative joint axis hypothesis that averages the configurations of multiple MPHs and lowers the noise from individual hypotheses.

After defining the joint axis orientation and the x- and y-coordinates of the position, the MPHs $h_i \in \chi(\mathcal{C}^{\text{a}})$ are used to determine the faces of the cuboid representing the door panel. This step estimates the dimensions and layout of the door panel by clustering distances to supporting planes, where each plane corresponds to a face of the door.

The supporting plane of the front face of the cuboid is defined by the planar patch used to generate the MPH. The supporting planes of the top and bottom faces, shown in

Figure 2.3, are assumed to be parallel to the plane Π. Therefore, these planes are defined by their distances from Π. Let d_i^{t} and d_i^{b} be the distances of the top and bottom faces of the cuboid corresponding to the MPH h_i. These distances can be calculated from the center and the size of this cuboid, as follows:

$$d_i^{\mathrm{t}} = z^\top t_{B,i} + \frac{l_{z,i}}{2}, \tag{2.8}$$

$$d_i^{\mathrm{b}} = z^\top t_{B,i} - \frac{l_{z,i}}{2}, \tag{2.9}$$

where z is the z-axis of S_A, $t_{B,i}$ is the translation vector of $S_{B,i}$ and $l_{z,i}$ is the size of the cuboid of the MPH along the z-axis. The distances computed for all MPHs $h_i \in \chi\left(\mathcal{C}^{\mathrm{a}}\right)$ are clustered using the clustering algorithm described in Section 2.3.1. The centers of the obtained clusters $\bar{d}^{\mathrm{t}}$ and $\bar{d}^{\mathrm{b}}$ define the positions of the top and bottom faces of the door hypothesis. At the same time, the midpoint $\frac{1}{2}(\bar{d}^{\mathrm{t}} + \bar{d}^{\mathrm{b}})$ defines the z-coordinate of the joint axis origin, placing the axis at the vertical center of the estimated door panel.

The supporting planes of the inner and outer faces of the cuboid representing the door panel are parallel to the joint axis and perpendicular to the front face. Therefore, these planes are defined by their distances from the joint axis. The outer face of the cuboid representing the door panel is calculated by clustering the distances d_i^{o} of the outer faces of the MPHs $h_i \in \chi\left(\mathcal{C}^{\mathrm{a}}\right)$, analogous to the calculation of the top and bottom faces. In the current implementation of the algorithm, the inner face of the door is assumed to lie directly on the joint axis, and its distance to the joint axis is set to 0.

From each MPH $h_i \in \chi\left(\mathcal{C}^{\mathrm{a}}\right)$, a pair (k, φ) is added to the sequence Θ, where k is the image index of the hypothesis h_i and φ is the estimated door state in the image k. The angle φ represents the relative orientation computed as the angle between the x-axes of S_A and $S_{B,i}$.

The proposed hierarchical clustering procedure is represented by Algorithm 2. The input of the algorithm is the set of all MPHs generated from an image sequence χ_{all} and the output is a set of door hypotheses $\mathcal{H}$ obtained by integrating the MPHs over the image sequence.

Drawer Hypotheses

To generate drawer hypotheses, the method first performs clustering based on the x-axes of the moving part hypotheses (MPHs), which correspond to the direction of motion. Similarly to the door hypotheses, the method first pairs MPHs to create x-axis candidates. For each pair of MPHs $\left(h_i, h_j\right)$, a symmetric distance matrix $E_x\left(i,j\right)$ is computed, similar to Equation (2.4):

$$E_x\left(i,j\right) = 1 - |x_i^\top x_j|, \tag{2.10}$$

where x_i and x_j correspond to x-axes of h_i and h_j, respectively. Using the computed matrix, the clustering procedure described in Section 2.3.1 groups together x-axis candidates with a similar orientation. Each resulting cluster $\mathcal{C}^{\mathrm{x}}$ of x-axis directions corresponds to a candidate drawer motion direction.

Let $\chi\left(\mathcal{C}^{\mathrm{x}}\right)$ be the set of MPHs involved in the formation of a cluster $\mathcal{C}^{\mathrm{x}}$. For each such cluster, a secondary clustering step is applied to the z-axes of the MPHs. To group hypotheses with similar front face orientations, a pairwise distance matrix $E_{\varphi,z}$ is computed for each pair of MPHs $\left(h_i, h_j\right) \in \chi\left(\mathcal{C}^{\mathrm{x}}\right)$:

$$E_{\varphi,z}\left(i,j\right) = 1 - \max\left(|\cos\varphi_{ij}|, |\sin\varphi_{ij}|\right), \tag{2.11}$$

Algorithm 2 Generating door hypotheses by hierarchical clustering

```
1: procedure IntegrateDoor($\chi_{\text{all}}$)
2:     $\mathcal{H} \leftarrow \emptyset$
3:     Create z-axis clusters $\mathcal{C}^{\text{z}}$.
4:     for every cluster $\mathcal{C}^{\text{z}}$ do
5:         Compute z-axis and plane $\Pi$.
6:         Create top face clusters $\mathcal{C}^{\text{t}}$.
7:         Compute center $\bar{d}^{\text{t}}$ of each cluster $\mathcal{C}^{\text{t}}$.
8:         Create bottom face clusters $\mathcal{C}^{\text{b}}$.
9:         Compute center $\bar{d}^{\text{b}}$ of each cluster $\mathcal{C}^{\text{b}}$.
10:        Compute MPPIs.
11:        Create joint axis clusters $\mathcal{C}^{\text{a}}$.
12:        for every cluster $\mathcal{C}^{\text{a}}$ do
13:            Compute $p$ and $r_x$.
14:            Create outer face clusters $\mathcal{C}^{\text{o}}$.
15:            Compute center $\bar{d}^{\text{o}}$ of each cluster $\mathcal{C}^{\text{o}}$.
16:            for every $\bar{d}^{\text{o}}$ do
17:                for every $\bar{d}^{\text{b}}$ do
18:                    for every $\bar{d}^{\text{t}}$ do
19:                        Generate a door hypothesis $H$.
20:                        Put $H$ into $\mathcal{H}$.
21:                    end for
22:                end for
23:            end for
24:        end for
25:    end for
26:    return $\mathcal{H}$
27: end procedure
```

where φ_{ij} is the angle between z_i and z_j of $S_{B,i}$ and $S_{B,j}$, respectively. This metric assigns lower values to MPHs whose z-axes are either parallel ($|\cos\varphi_{ij}| = 1$) or orthogonal ($|\sin\varphi_{ij}| = 1$). By doing that, it captures the amount of rotational misalignment about the x-axis, allowing the method to group MPHs with consistent front face orientations into clusters $\mathcal{C}^{\text{z}}$, which represent joint axis candidates.

For each cluster $\mathcal{C}^{\text{z}}$, the method computes distances of the top, bottom, left, and right faces of the MPH's bounding box with respect to S_B, by projecting the center of the box and adding or subtracting half the size along the relevant axis, similar to the door hypotheses. These distances are then clustered across MPHs to identify consistent positions of the drawer's faces.

To identify the zero-state of the drawer, the hypothesis whose origin has the smallest x-coordinate in RF S_A obtained by the hierarchical clustering procedure is selected. Then, for each MPH $h_i \in \chi(\mathcal{C}^{\text{z}})$, a pair (k, ω) is added to the sequence Θ, where k is the image index of the hypothesis h_i, and ω is the state of the drawer computed as the displacement along the x-axis from the default position. The drawer hypothesis generation process is summarized in Algorithm 3.

2.3.4 Hypothesis Evaluation and Selection

Given an input image sequence $\mathcal{I}$, a set $\mathcal{H}$ of door and drawer hypotheses is generated. Let s denote the state parameter, where $s = \varphi$ represents the opening angle for a door

Algorithm 3 Generating drawer hypotheses by hierarchical clustering

```
1: procedure IntegrateDrawer($\chi_{\text{all}}$)
2:     $\mathcal{H} \leftarrow \emptyset$
3:     Create x-axis clusters $\mathcal{C}^{\text{x}}$.
4:     for every cluster $\mathcal{C}^{\text{x}}$ do
5:         Create z-axis clusters $\mathcal{C}^{\text{z}}$.
6:         for every cluster $\mathcal{C}^{\text{z}}$ do
7:             Select default state with smallest $x$-coordinate.
8:             Create face clusters $\mathcal{C}^{\text{t}}, \mathcal{C}^{\text{b}}, \mathcal{C}^{\text{l}}, \mathcal{C}^{\text{r}}$.
9:             Compute centers $\bar{d}^{\text{t}}, \bar{d}^{\text{b}}, \bar{d}^{\text{l}}, \bar{d}^{\text{r}}$ of each cluster.
10:            for every $\bar{d}^{\text{l}}$ do
11:                for every $\bar{d}^{\text{r}}$ do
12:                    for every $\bar{d}^{\text{t}}$ do
13:                        for every $\bar{d}^{\text{b}}$ do
14:                            Generate drawer hypothesis $H$.
15:                            Put $H$ into $\mathcal{H}$.
16:                        end for
17:                    end for
18:                end for
19:            end for
20:        end for
21:    end for
22:    return $\mathcal{H}$
23: end procedure
```

and $s = \omega$ the extension for a drawer. Each hypothesis $H \in \mathcal{H}$ is evaluated by computing its *hypothesis evaluation score*:

$$\Psi\left(H,k\right) = \zeta\left(H,k\right) \cdot \left(\Omega\left(H,k\right) - \left|\Phi\left(H,k\right)\right|\right) \tag{2.12}$$

where Ω denotes the *scene fitting score* and $|\Phi|$ the *transparency cost*, as proposed in Cupec et al., 2020, while ζ represents the *zero-state factor*. The zero-state factor is defined as:

$$\zeta\left(H,k\right) = 1 + e^{-(s/\sigma)^2}, \tag{2.13}$$

which assigns greater weight to hypotheses that are close to the zero-state, because it is a priori more likely than other states. The value σ is a user-defined constant set to 5° for doors and 0.05 m for drawers.

The score Ψ is computed for each pair (k, s) in the state sequence Θ associated with a hypothesis H. For a given hypothesis H and state s, the pose of the cuboid (representing the door panel or drawer front) with respect to the camera frame S_C is computed using the transformation ${}^{C}T_{A}$, the offset vector r, and the corresponding state parameter s. The resulting pose is denoted by ${}^{C}T_{B}(s)$ and is obtained as:

$${}^{C}T_{B}(s) = {}^{C}T_{A}\,{}^{A}T_{B}(s), \tag{2.14}$$

where ${}^{A}T_{B}(s)$ is the pose of the frame S_B with respect to S_A for the state s.

The surface of the cuboid is uniformly sampled and the obtained point cloud is transformed into the scene using ${}^{C}T_{B}(s)$. By comparing this transformed point cloud with the scene point cloud computed from the k-th depth image of the input image sequence, $\Omega\left(H,k\right)$ and $\Phi\left(H,k\right)$ are computed as described in Cupec et al., 2020.

Hypothesis Temporal Tracking

To evaluate and refine the hypotheses $\mathcal{H}$ over time, each hypothesis is further analyzed through a temporal tracking procedure, resulting in a refined set $\mathcal{H}'$. First, for each $H \in \mathcal{H}$, the earliest and latest image indices in the sequence $\mathcal{I}$ that contributed to the H are identified, defining the temporal window $[k_{\text{min}}, k_{\text{max}}]$. This window is used to form a set of temporally consistent subsequences of states Θ^*. Each Θ^* represents a sequence of states in which consecutive states differ by less than a predefined threshold τ_t. The subsequences Θ^* are initialized from MPH states that contributed to H at k_{min}. From there, they are extended iteratively by including subsequent states that are within the threshold τ_t. If a state does not satisfy this consistency condition with any existing subsequence, a new subsequence is created starting from that state.

In cases where multiple valid states occur at the same time index k for a given subsequence, the sequence is duplicated, and each copy is extended with one of the candidate states. This mechanism allows multiple candidate subsequences to be formed in parallel. Once all candidate subsequences are generated, each is scored by summing the hypothesis evaluation scores Ψ of its states:

$$\Psi_\Sigma\left(H, \Theta^*\right) = \sum_{(k,s) \in \Theta^*} \Psi\left(H, k\right). \tag{2.15}$$

For each hypothesis H, the subsequence Θ^* with the highest score is selected and used to construct a refined hypothesis H', retaining only the states from the chosen subsequence.

An additional refinement step is applied to each hypothesis in $\mathcal{H}'$ to reduce the influence of the hypothesis evaluation score Ψ obtained in the initial frames. If the object is a part of a larger structure (e.g. a closet with multiple doors and/or drawers), misaligned hypotheses may appear more likely due to high evaluation scores Ψ in the initial frames, as the scene fitting score Ω may incorrectly include other parts of the structure as part of the object. To mitigate this, each state $s(k)$ in the subsequence Θ^* is compared to the initial state $s(k_{\text{min}})$. If the absolute difference $e_s = |s(k) - s(k_{\text{min}})|$ is smaller than a predefined threshold τ_s, that state does not contribute to the total score. This results in an adjusted total score:

$$\tilde{\Psi}_\Sigma\left(H', \Theta^*\right) = \Psi_\Sigma\left(H', \Theta^*\right) - \sum_{(k,s) \in \Theta^*} S_e\left(e_s\right) \cdot \Psi\left(H', k\right), \tag{2.16}$$

where

$$S_e(e_s) = \begin{cases} 1, & \text{if } e_s \leq \tau_s, \\ 0, & \text{otherwise.} \end{cases}$$

In addition, to filter out background objects that remain stationary, any hypothesis whose total change in state across the sequence does not exceed a user-defined threshold is discarded. Among the refined set $\mathcal{H}'$, the hypothesis H^* with the highest total score $\tilde{\Psi}_\Sigma$ is selected as the final result.

Zero-State and Opening Direction Selection

To define the motion of the selected hypothesis H^* over time, the initial state $s(k_{\text{min}})$ in its subsequence Θ^* is defined as the zero-state. All other states in Θ^* are then expressed relative to this zero-state by computing the difference between the state at each index k and $s(k_{\text{min}})$.

For doors, the opening direction μ is determined by summing the angular states φ across the entire sequence Θ^*. If the sum is positive, the door is considered to open in the positive direction ($\mu = 1$). Otherwise, it opens in the negative direction ($\mu = -1$). For drawers, the direction is not explicitly estimated, as drawer motion is always assumed to occur along the positive x-axis.

Based on the final hypothesis H^*, a DM is created and inserted into the augmented environment map. The pose of this DM with respect to the environment map RF S_W is calculated by:

$$^{W}T_A = {}^{W}T_C \cdot {}^{C}T_A,$$

where $^{W}T_C$ is provided by the mobile robot localization system.

2.4 State Estimation

After the robot learns the location and movement direction of doors and drawers, it can estimate their states when it stops in front of them and captures an RGB-D image. The following procedure is used:

1: Predict the target object's location in relation to the camera.
2: Detect one or more planar patches within a region of interest (RoI).
3: Generate hypotheses about the moving part's state from the planar patches in the RoI.
4: Evaluate the hypotheses and select the most likely one.
5: Compute the object state, which is the angle of the door panel or drawer extension relative to the zero-state.

It is assumed that the robot localization system can provide information about the camera pose with respect to the environment map RF S_W in the form of an HTM $^{W}T_C$. The augmented environment map described in Section 2.2 contains the information about the doors and drawers in the robot's environment in the form of DMs, where each DM is associated with a matrix $^{W}T_A$ describing the pose of a door or a drawer with respect to S_W. The matrices $^{W}T_C$ and $^{W}T_A$ can be used to compute $^{C}T_A$ defining the pose of a particular DM with respect to the camera RF S_C.

An accurate estimate of the state of a door or a drawer requires an accurate pose $^{C}T_A$. Since this pose is computed using the camera pose $^{W}T_C$ provided by the robot localization system, an error in the robot localization will result in an error in the state estimate. To compensate for the inaccuracy of the robot localization, the local environment model is stored, i.e. a set of planar patches in the vicinity of a door or a drawer detected by the DDD-THD algorithm. This is used to align the query image with this local environment model. An ICP-based algorithm can be used for this alignment, using the pose $^{W}T_C$ provided by the robot localization system as the initial solution. In the experiments, an algorithm specifically designed for aligning rectangular structures is used, but this is beyond the scope of this thesis and therefore not described. An example of a robot capturing an object's state is shown in Figure 2.11.

Given the pose of a DM $^{C}T_A$, a Region of Interest (RoI) is computed in the form of a 3D bounding box aligned with the axes of S_A. The size of this bounding box in the y- and z-directions is equal to the second and third element of the size vector l, defined in Section 2.2, extended by 10%, while its size in the x-direction is a constant value large enough for the RoI to contain a door or a drawer in any state. In the current implementation, this value is set to 1.1 m.

For each planar patch with at least 1000 points within the RoI, a DH is generated with the x-axis of S_B parallel to the planar patch normal. The door state is computed as the angle between the x-axes of S_B and S_A and the drawer state is computed as the distance between the planar patch and the drawer front in the zero-state. Each DH is evaluated

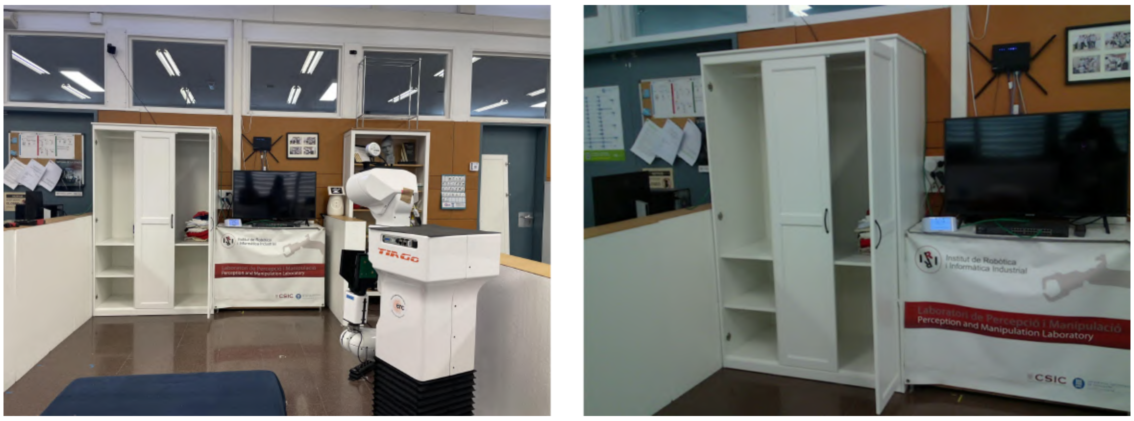
Figure 2.11: Example of a robot estimating the state of an articulated object. The left image shows the robot capturing the test object in a specific state, while the right image presents the corresponding view from the robot's camera.

using the approach described in Section 2.3.4, and all hypotheses with negative hypothesis evaluation scores are rejected. In some cases, a door panel is oriented in such a way that it is aligned with the camera optical rays, i.e. the angle between its surface and the camera optical rays pointing to that surface is very small. In such cases, the door panel is poorly visible in the image, and there is no suitable planar patch representing this surface. An example of this is shown in the bottom-right image of Figure 2.13. Thus, if no hypothesis with a positive hypothesis evaluation score is generated, the door panel is assumed to be aligned with the optical rays of the camera, and its state is calculated accordingly.

2.5 Experimental Evaluation

Two experiments were conducted to evaluate the proposed approach. The goal of the first experiment was to test the ability of DDD-THD to recognize doors or drawers in sequences of RGB-D images in which a human demonstrates opening a door or a drawer. The second experiment aimed to measure the accuracy of the state estimation achieved by DDD-SE. The test images were captured in a laboratory at the *Institut de Robòtica i Informàtica Industrial* in Barcelona, which replicated a typical household setting with furniture. Four different test objects were used in the experiments: a room door, a kitchen cabinet with two doors and one drawer, a nightstand with one door, and a wardrobe with two doors (see Figure 2.2).

For the experiments, a TIAGo robot mobile platform was utilized. This robot has the ability to autonomously create a map of its environment and navigate without colliding with obstacles by using the ROS Advanced Navigation Stack from PAL Robotics©. Additionally, it possesses a 7-DOF robotic manipulator attached to its chest, enabling it to grasp objects effectively. To improve its visual capabilities, an Intel RealSense LiDAR Camera L515 was installed on top of its head. This addition enables capturing highly detailed RGB-D images of the objects. This is particularly important for obtaining accurate state estimates.

2.5.1 Door/Drawer Detection

To test the accuracy of door/drawer detection, 38 sequences of RGB-D images were captured in which a human opens the doors or drawers of the test objects. The DDD-THD algorithm was then applied to these sequences. For each image sequence, DDD-THD generated a DM. In addition, the images of the four test objects were manually annotated by outlining the moving parts to obtain the ground truth. The ground truth annotations are compared to the image projections of the front faces of the DMs in the zero-state (closed doors and drawers) by computing Intersection over Union (IoU) as the detection performance index.

The results of the door and drawer detection experiment are summarized in Table 2.1. Of the 38 captured sequences, 6 belong to a room door, 12 to the left door of a wardrobe, 12 to the right door of the wardrobe, and 2 to each of the following moving parts: drawer, small cabinet door, large cabinet door and nightstand door. Overall, the proposed DDD-THD algorithm achieved a mean IoU of 83.62% across all tested objects. However, the detection accuracy varies across object categories. The room door and the wardrobe right door show the highest mean IoU values of 88.33% and 92.25%, respectively. These results suggest that the algorithm performs particularly well for large planar surfaces with clear visual boundaries. In contrast, the drawer achieved the lowest mean IoU of 70.70%, primarily due to occlusion during the demonstration. In one instance, the demonstrator's hand covered a large portion of the drawer's front surface, leading to an underestimation of its dimensions. This highlights a limitation of approaches utilizing demonstrations, as their performance is sensitive to the visibility of the manipulated part during the teaching phase.

The wardrobe left door also showed a slightly lower mean IoU of 81.59% compared to its right counterpart. This discrepancy can be attributed to the wardrobe's placement in the room. On the left side, a nearby wall restricted the demonstrator's movement, making it difficult to fully clear the view of the door surface during demonstrations and thus partially occluding it. Similarly, the nightstand achieved an IoU of 79.00%. This result is most likely due to its smaller size, compared to other objects in the dataset, which increased the likelihood of partial occlusion by the demonstrator's hand. Additionally, the generated model was more prone to including false points due to noise, which further reduced its accuracy.

Table 2.1 Intersection over Union for each moving part

Moving Part	**Min. IoU**	**Max. IoU**	**Mean IoU**
Room door	85.21%	94.52%	88.33%
Drawer	58.66%	82.74%	70.70%
Small cabinet door	78.94%	91.13%	85.04%
Large cabinet door	81.59%	89.57%	85.58%
Nightstand	78.79%	79.21%	79.00%
Wardrobe left door	72.58%	90.50%	81.59%
Wardrobe right door	84.91%	95.38%	92.25%

These findings demonstrate that the proposed approach can reliably detect a variety of articulated parts in realistic household scenarios, with only a slight performance drop for smaller or partially occluded objects. The qualitative results in Figure 2.12 further demonstrate that the detected bounding boxes (in red) closely align with the annotated ground truth (in green) across diverse object types.

2.5.2 State Estimation

The accuracy of the state estimation of the proposed approach was evaluated by applying the DDD-SE algorithm to RGB-D images of the considered test objects in different states and comparing the estimated state values with manually measured ground truth

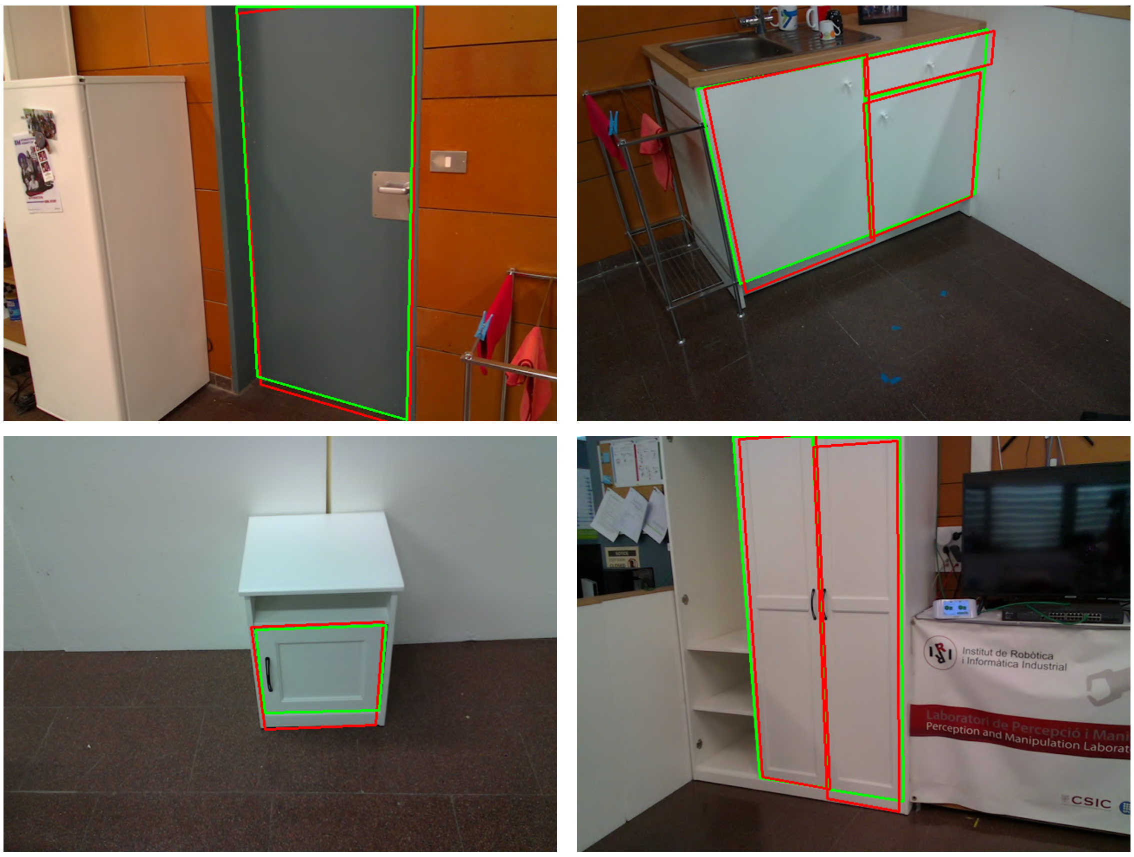
Institut de Robòtica i Informàtica Industrial
Perception and Manipulation Laborat
CSIC

Figure 2.12: Door/drawer detection results for the test objects. The ground truth data is shown with a green bounding box, while the detected parts are shown with red bounding boxes.

data. The results of this experiment are summarized in Table 2.2, which reports the total number of occurrences, the number and percentage of correct measurements, and the average absolute error for each moving part of the test objects.

A measurement was considered correct if the absolute difference between the estimated and ground-truth value was at most 5° for doors or 0.05 m for the drawer. These thresholds were obtained by a camera measurement error analysis. Specifically, a scene with a dominant planar surface was analyzed by fitting a plane to the points on that surface detected by the 3D camera and calculating the standard deviation of those points from the plane. This standard deviation was $\varepsilon = 0.0068$ m. Assuming a normal distribution of the camera measurement error, almost all points on the surface under consideration are within 3ε of the plane, which was taken as the maximum error in plane estimation.

The state of the drawer is measured by calculating the distance between the drawer front and the static part of the furniture to which this drawer belongs. If the worst case is assumed, where both the drawer front and the furniture front are estimated with the maximum error of 3ε, then the error in the state measurement is $6\varepsilon = 0.0408$ m. Rounding up this value to the next integer number in centimeters, the tolerance of $\tau_d = 0.05$ m is obtained. The tolerance for the door state was determined by assuming that the maximum error in measuring the distance between the point of the door panel farthest from the door axis and the static part of the furniture is 6ε. Assuming a reference door width of $w_{\text{door}} = 0.5$ m, the door angle corresponding to this distance is $6\varepsilon / w_{\text{door}} = 4.68°$. The nearest integer value in degrees is used as the tolerance for the door state $\tau_\varphi = 5°$.

The results presented in Table 2.2 show that the proposed method achieves high accuracy overall. The drawer achieved the best performance, with 98.72% correct estimates and a mean error of only 6 mm. This can be explained by the fact that drawer motion corresponds to a simple linear translation, which is well captured by depth sensing. The large cabinet door also showed excellent results, with all 78 estimates correct and an average error of 0.53°.

In contrast, performance was lower for the room door and the nightstand. The room door achieved only 63.33% correct estimates, despite having a relatively low mean error of 1.35°. This is mainly due to challenging view angles: when the door is oriented such that the camera rays strike at a shallow angle, the surface becomes difficult to reconstruct in the depth image. Similarly, the nightstand obtained a lower accuracy of 70.00% with an average error of 2.70°. Here, the relatively small dimensions of the door, combined with specular reflections from its surface, reduced the reliability of the depth measurements. In the case of the small cabinet door, there is a wall to the right of the cabinet that is a flat vertical surface barely distinguishable from the door panel when it is open at 90°, which is another cause of incorrect estimates. An example of such a case is shown in Figure 2.13 (bottom right), where the misidentified door panel is outlined in red.

Furthermore, since the proposed method relies on the alignment of furniture surfaces and walls, it fails in cases where the scene has a simple geometry with few surfaces available for matching and some of these surfaces cannot be reconstructed by the RGB-D camera used due to specular reflection. This is the cause of the lower performance of the proposed algorithm for the nightstand shown in Figure 2.13 (bottom left).

Table 2.2 State estimation accuracy for each moving part

Moving part	Total	Correct	%	Avg. Error
Room door	30	19	63.33	1.35°
Drawer	78	77	98.72	6.0 mm
Small cabinet door	78	67	85.90	0.71°
Large cabinet door	78	78	100.00	0.53°
Nightstand	30	21	70.00	2.70°
Wardrobe left door	54	43	79.63	0.61°
Wardrobe right door	54	44	81.48	1.74°

2.6 Conclusion

This chapter presented a novel framework for detecting and modeling doors and drawers from human demonstrations, with state estimation from a single RGB-D image. By observing a person opening a door or a drawer, the system generates models of the articulated parts and incorporates them into an environment map. The approach enables the robot to determine both the location of these objects and their kinematic states from subsequent observations. The method presented in this chapter was published in Cupec et al., 2023.

The experimental results demonstrated reliable detection and accurate state estimation across a variety of furniture types, confirming that demonstrations provide an effective source of object information. Certain degenerate cases were identified, particularly when the plane of the door is nearly parallel to the camera axis and thus difficult to reconstruct from depth data. Such situations could be mitigated by adopting next-best-view strategies to reposition the robot for improved visibility, or by integrating RGB-based methods in cases where no depth information is available. Furthermore, another

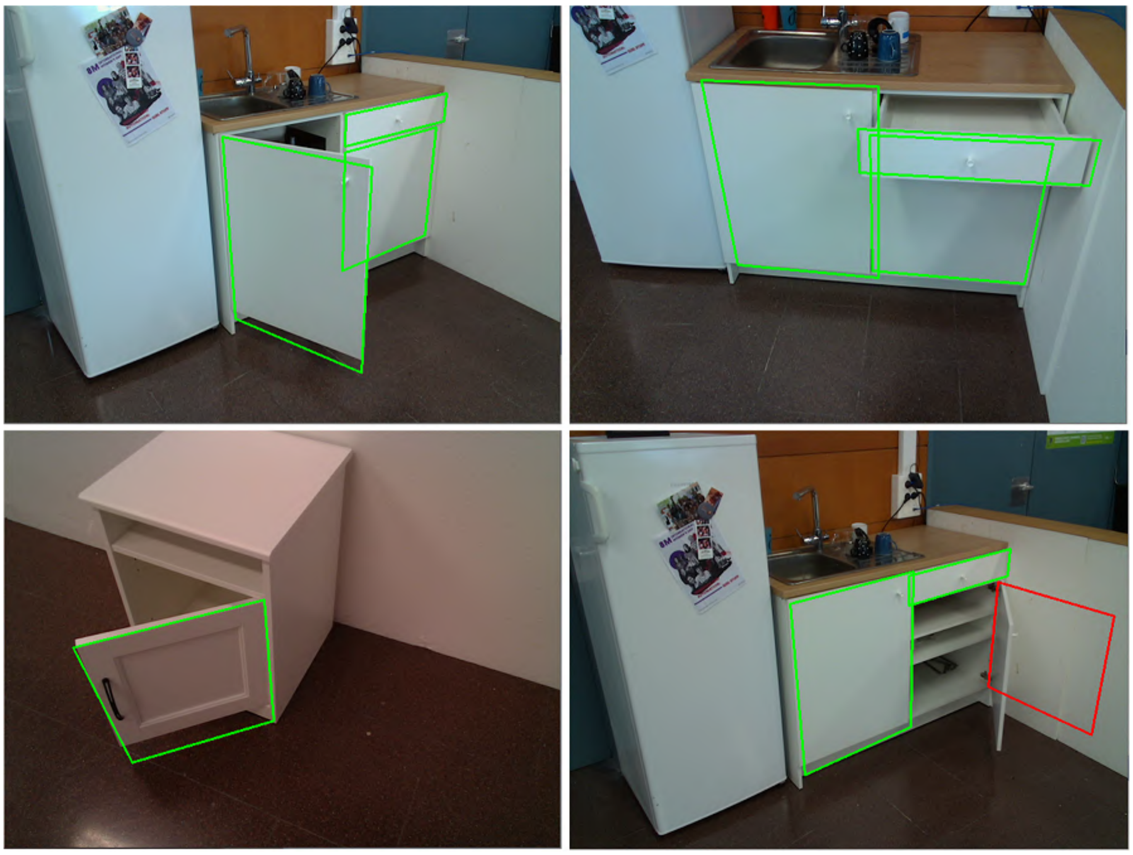
Figure 2.13: Examples of correct (green) and incorrect (red) state estimates.

solution could include machine learning models trained on images of doors in multiple states captured from different viewpoints, ensuring that at least one view clearly reveals the door surface. Other interesting future work avenues include enhancing the robot's physical interaction capabilities with its environment, particularly in assistive scenarios such as locating specific household objects.

Alongside these contributions, this framework provides a basis for the motion planning and interaction strategies presented in the following chapters, where the robot plans and executes manipulation tasks on doors based on the door models presented in this chapter.

3 Multi-Contact Door Opening

Building on the framework presented in the previous chapter, which enables the detection and kinematic modeling of doors and drawers, this chapter focuses on planning and executing the corresponding manipulation actions for opening furniture doors.

In this process, motion planning plays a pivotal role, generating feasible trajectories for robot manipulators, considering constraints such as kinematics, dynamics, and environmental obstacles (LaValle, 2006; Gammell and Strub, 2021; Mukadam et al., 2018). These algorithms aim to produce collision-free paths in the robot's configuration space, facilitating smooth and precise manipulation.

While numerous motion planning methods for door opening exist, many are tailored to scenarios involving doors equipped with handles (Nagatani and Yuta, 1996; Karayiannidis et al., 2016; Arduengo, Torras and Sentis, 2021). However, in real-world environments, furniture doors often lack handles, presenting a unique set of challenges for robotic manipulation. Existing approaches for door opening typically rely on single-point contact strategies, such as pushing or pulling at specific locations on the door. Nevertheless, such methods may be inadequate for handling the diverse configurations and properties of furniture doors, and they fail to fully utilize the robot's workspace to explore more versatile and robust manipulation strategies.

To address this limitation, this chapter proposes a multi-contact robot arm path planning framework for opening handleless furniture doors. The proposed approach enables robots to interact with furniture doors using multiple points of contact, improving robustness and expanding the set of manipulation actions. Furthermore, the proposed method is comprehensively evaluated through both simulation and real-world experiments, demonstrating its robustness across a variety of furniture door configurations and properties. The results highlight the potential of multi-contact path planning to enhance robotic manipulation capabilities in human-centric environments, particularly in the context of furniture door handling. Moreover, the proposed method can be applied to any door that is partially opened.

A commonly studied scenario for opening doors with robots is that of a mobile manipulator that uses its movement capabilities to adjust the position of the robot arm base to accomplish a given task. In this chapter, the problem of planning paths for a robot arm that cannot move its base with respect to the target door is considered, as shown in Figure 3.1. Although service robots in households are expected to be mobile, the considered problem is important because it may occur in cluttered environments where the robot movement is restricted by obstacles. In such cases, the proposed multi-contact path planning method can be useful because it significantly expands the space of feasible trajectories that can solve a given task compared to the single-contact fixed-grasp planning methods.

3.1 Related Work

Door opening is a widely studied topic in robotic manipulation, particularly due to its relevance in service and assistive robotics. While previous research has investigated a

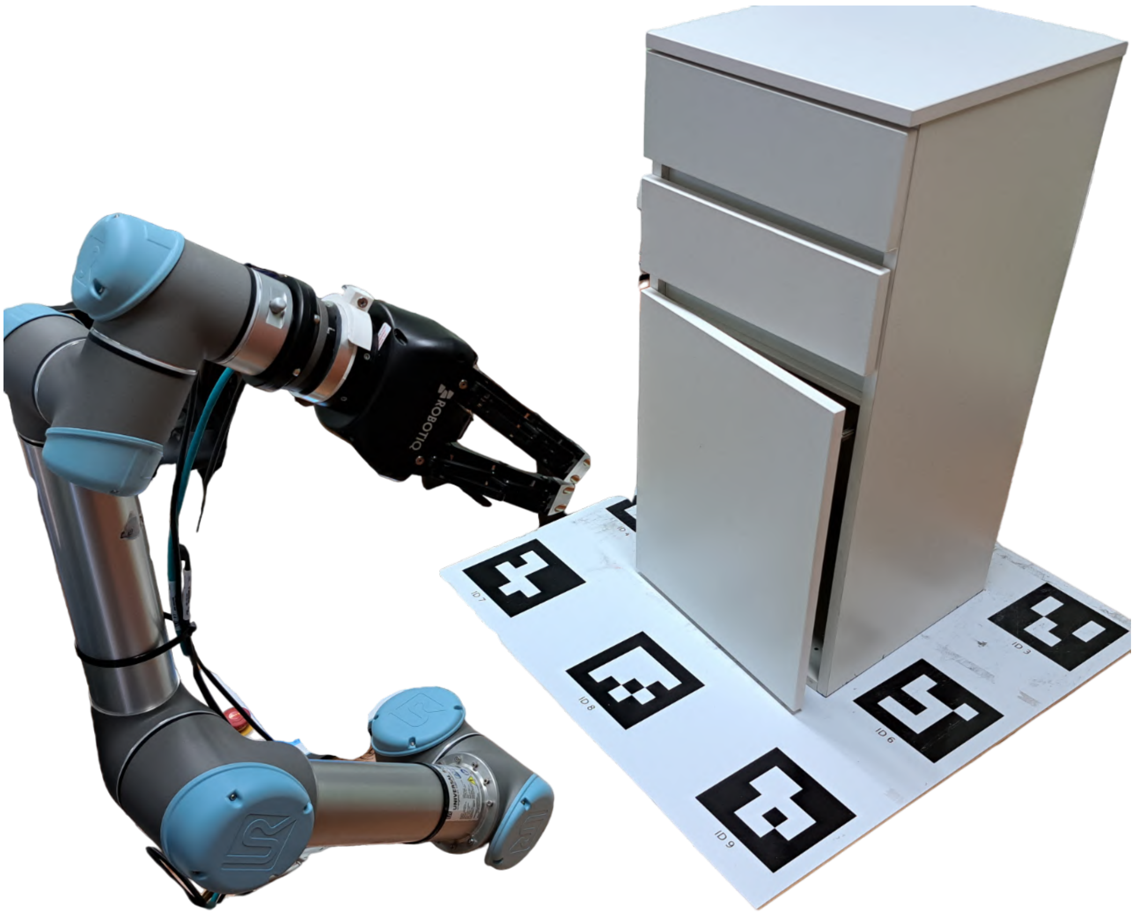

Figure 3.1: Real-world experimental setup featuring a cabinet with handleless doors and a push-latch opening mechanism, operated by the Universal Robots UR5 robotic arm equipped with the Robotiq 3-Finger Gripper. The ArUco board was used to ensure proper hand–eye calibration.

range of strategies for robotic interaction with doors, the majority of approaches have focused on single-contact methods in order to open them. This section reviews key works related to general motion planning and door-opening techniques, with a focus on methods relevant to the approach presented in this chapter.

3.1.1 Motion Planning

Sampling-based and trajectory optimization methods have become the two most common approaches to high-dimensional robot motion planning. Sampling-based methods create and iteratively refine a graph or a tree structure representing connectivity between feasible robot configurations. Most methods are probabilistically complete (LaValle, 2006), i.e., the probability of finding a solution approaches one as more time is spent. Sampling-based methods can be categorized as single-query, such as Rapidly-Exploring Random Trees (RRT) (LaValle, Kuffner, Donald et al., 2001), Expansive Space Trees (EST) (Hsu, Latombe and Motwani, 1997) and their variants (Kuffner and LaValle, 2000; Karaman and Frazzoli, 2011; Salzman and Halperin, 2016; Li, Littlefield and Bekris, 2016; Shen, Xie and Zhu, 2024), or as multi-query, notably Probabilistic Roadmaps (PRM) (Kavraki et al., 1996) and its variants (Bohlin and Kavraki, 2000; Karaman and Frazzoli, 2011). Some methods are asymptotically optimal, implying that they converge to the optimal solution as the number of samples goes to infinity (Karaman and Frazzoli, 2011; Gammell and Strub, 2021). The mentioned methods can efficiently find a feasible path in the high-dimensional configuration space, which could be used to plan the robot's motion toward the contact point of the door. However, they are not suitable for planning the motion for door opening, since the door movement implies that the robot's environment is changed at every time instant, which is difficult to encode as a cost or constraint in sampling-based methods. While the door could be modeled as a part of the kinematic chain, in that case only a single contact point between the end-effector and the door would be considered, which may be too restrictive and inhibit efficient planning.

Trajectory optimization methods try to find a sequence of states that minimizes a user-defined cost function, which usually encodes smoothness or energy expenditure, while adhering to defined constraints, such as collision avoidance and kinematic limits of the robot. Trajectory optimization methods can be categorized as gradient-based or stochastic optimization-based. Gradient-based trajectory optimization methods, such as CHOMP (Zucker et al., 2013), TrajOpt (Schulman et al., 2014) and GPMP2 (Mukadam et al., 2018), are computationally very efficient even for high-dimensional motion planning problems, but they are prone to infeasible local minima. On the other hand, stochastic trajectory optimization methods, such as STOMP (Kalakrishnan et al., 2011b), cross-entropy motion planning (Kobilarov, 2012) and MGPTO (Petrović, Marković and Petrović, 2022), can handle non-differentiable cost functions and constraints and more often find feasible solutions in complex environments, albeit with a higher computational cost. The STOMP update rule was applied for improving learned force control policies for manipulation with contact and the method was deployed for opening a door with a handle (Kalakrishnan et al., 2011a). None of the mentioned trajectory optimization methods have been utilized for opening handleless doors or reasoning about multiple contact points between the end-effector and the door.

3.1.2 Door Opening Manipulation

Using robotic arms for opening and closing doors has been an active area of research in the past three decades (Nagatani and Yuta, 1996; Peterson, Austin and Kragic, 2000; Chung et al., 2009; Jain and Kemp, 2010; Rühr et al., 2012). Many methods target the problem of push-opening a door with a handle, where the challenge is in the appropriate task sequence planning. After the door handle is detected and grasped, the robot holds it while pushing the door with a circular arm motion. One of the earliest works in push-opening doors (Nagatani and Yuta, 1996) utilized action primitives control using sequences of planned motion primitives. Using a state machine for task planning was proposed in Meeussen et al., 2010, where configuration space paths were planned with a grid-search algorithm (Chitta, Cohen and Likhachev, 2010), while a compliant control scheme was implemented for trajectory execution. Stuede et al., 2019 proposed a control structure that uses the handle frame for reference in an outer loop for the robot arm Cartesian impedance controller. The mentioned works provide valuable insights for handle grasping and sequence planning but are not applicable for opening furniture doors in a household environment, which usually require pulling.

Many works thus focus on pull-opening doors with a handle, which presents a more challenging manipulation problem due to requiring collision avoidance while moving the door toward the base of the robot arm. An early work (Peterson, Austin and Kragic, 2000) employed force/torque control algorithms for pull-opening a door with a mobile manipulator. Impedance control frameworks (Jain and Kemp, 2010; Rühr et al., 2012) were proposed to robustly pull-open a variety of doors and drawers in household environments. Authors in Nemec, Žlajpah and Ude, 2017; Ding et al., 2021 proposed intelligent robot control based on reinforcement learning (RL) to open different types of pull doors without previous knowledge of the environment. Karayiannidis et al., 2012; Karayiannidis et al., 2016 proposed adaptive force/velocity control under uncertainty that achieves robust door manipulation by simultaneous compliant interaction and estimation of constraints imposed by the joint. Arduengo, Torras and Sentis, 2021 presented a complete perception, planning and control framework that relies on the Task Space Regions (TSR) algorithm (Berenson, Srinivasa and Kuffner, 2011) for pose-constrained manipulation planning. Pull-opening a door was achieved by formulating it as a nonlinear optimization problem (Kuribayashi and Yamazaki, 2023), utilizing differential algebraic equations for robot modeling. Mittal et al., 2022 formulated the door-opening control as a holistic optimization problem, where the control of both the base and the arm is integrated with Euclidean signed distance field environment representation to navigate around obstacles while executing manipulation tasks. The performance of mobile manipulators in pull-opening doors was shown to improve by combining quadratic programming with Cartesian compliance control (Thamrongaphichartkul and Vongbunyong, 2024). Recently, three frameworks able to open both pull and push-type doors were proposed, where Kang et al., 2024a combined adaptive position-force control and deep RL, while Jang, Kim and Park, 2023 first used a graph search algorithm to plan the mobile base path and then compute the robot arm inverse kinematics that achieves the desired door angle. Wang et al., 2020 propose learning semantic 3D keypoints as door handle representations to generate the end-effector trajectory from a motion planner.

The mentioned methods rely on grasping the door handle before pushing or pulling, which can be restrictive in real-world environments due to the kinematic limitations of the robot and the perception uncertainty. Moreover, different types of doors do not have a handle, such as those using the push-latch mechanism, requiring different approaches to door manipulation. Finally, methods based on reinforcement learning generally require retraining when applied to different robots, whereas the presented approach was designed to be applicable to any robot without retraining.

A framework for estimating door dynamics was proposed in Endres, Trinkle and Burgard, 2013, where the door is forcefully pushed according to the estimated model and opening is achieved without a continuous hold of the handle. However, the method can only be used for push-opening and the door state is not continuously controlled by the robot. Axelrod and Huang, 2015 proposed a pipeline where the door is first unlatched by pushing the handle, followed by placing the robot arm between the door and the door frame and then pulling or pushing in an open loop manner. Prats, Sanz and Del Pobil, 2010 proposed non-prehensile door opening by combining tactile information with vision and force feedback, but the method is suitable only for sliding doors. To the best of current knowledge, there are no methods that reason about multiple contact points between the robot gripper and the door. All described methods consider only a single contact point, which can shrink the collision-free configuration space due to kinematic limitations and make the motion planning problem infeasible.

3.2 Problem Definition

3.2.1 Furniture Model

In this chapter, a piece of furniture with a door is considered and modeled as described in Chapter 2. To extend this model, two additional reference frames (RFs) are introduced, as illustrated in Figure 3.2: RF S_F, a frame fixed to the furniture, and RF S_D, a frame located at the back vertex of the door panel. The door panel size vector l, the position of the panel relative to the joint axis r, the axis-side parameter μ and the door state φ are as defined in Section 2.2.1. Let $S_{A'}$ be the RF resulting from the rotation of joint axis frame

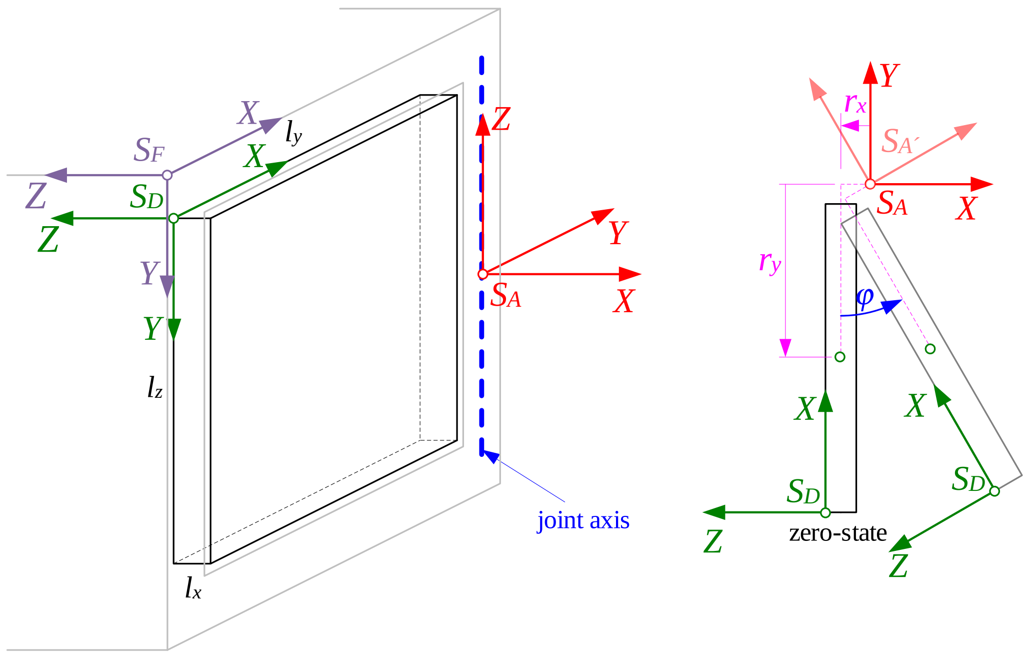

Figure 3.2: Reference frames of the proposed furniture door model. The frame S_F (purple) is fixed to the furniture, S_D (green) is attached to the back corner of the door panel, and $S_{A'}$ is obtained by rotating the reference frame S_A around its z-axis.

S_A around its z-axis by the angle φ (see Figure 3.2). Then

$$^{A}T_{A'}(\varphi) = \begin{bmatrix} R_z(\varphi) & 0 \\ 0 & 1 \end{bmatrix}, \tag{3.1}$$

where $R_z(\varphi)$ describes the rotation around the z-axis by the angle φ.

The origin of S_D lies on the back face of the door and its x- and y-axes lie in the supporting plane of this surface. The relation of S_D with respect to $S_{A'}$ in the closed state is described by:

$$^{A'}T_D = \begin{bmatrix} 0 & 0 & -\mu & r_x - \mu\frac{l_x}{2} \\ \mu & 0 & 0 & r_y - \frac{l_y}{2} \\ 0 & -1 & 0 & \frac{l_z}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

A world RF S_W can be positioned anywhere in the robot's environment, as described in Chapter 2. It is assumed that the pose of S_A with respect to the world RF S_W is known and that it is represented by the homogeneous transformation matrix $^{W}T_A$. Then, for a given angle φ, the pose of S_D with respect to S_W can be computed by

$$^{W}T_D(\varphi) = {}^{W}T_A\, {}^{A}T_{A'}(\varphi)\, {}^{A'}T_D. \tag{3.3}$$

The door parameters required for planning a door-opening path can be represented by a tuple $\mathcal{D} = \left({}^{W}T_A, l, r, \mu\right)$. The volume of space in the robot's environment that is occupied by the furniture and the door is denoted by $\mathcal{E}$, where $\mathcal{E}_s$ is the volume occupied by the static part of the furniture and $\mathcal{E}_d$ is the volume occupied by the door panel.

3.2.2 Path Planning Problem

The problem addressed in this chapter is the planning of a path for a robotic manipulator that opens handleless doors by pressing the back face of the door panel with its tool. This problem can be defined as a search for a path Q that represents a sequence of robot joint configurations:

$$Q = (q_0, q_1, \ldots, q_{n-1}), \tag{3.4}$$

where q_k stands for joint vectors (configurations). Each configuration q_k of the planned path corresponds to a door state, i.e. an angle φ_k, and must fulfill the following *state conditions*:

1. The configuration q_k must be reachable taking into account the kinematic constraints and the joint limits of the robot.
2. There must be no collision between the robot and the static part of the furniture. This can be formulated by
$$\mathcal{R} \cap \mathcal{E}_s = \emptyset, \tag{3.5}$$
where $\mathcal{R}$ is the volume occupied by the robot.
3. There must be no collision between the robot and the door panel:
$$\mathcal{R} \cap \mathcal{E}_d = \emptyset. \tag{3.6}$$
4. The point of contact between the tool and the panel surface can only be on a certain predefined part of the tool surface, which is referred to in this chapter as the *tool contact surface*. An example of a tool contact surface is shown in Figure 3.3. The orthogonal projection of the tool contact surface must overlap sufficiently with the back face. Let C_S be the set of points that represent the contact surface. The tool contact surface is considered sufficiently overlapping with the back face if
$$\exists P \in C_S, \quad \min\left\{{}^D p_x, {}^D p_y\right\} \geq \tau_{eL}, \tag{3.7}$$
where ${}^D p_x$ and ${}^D p_y$ are the x- and y-coordinates of a point P with respect to RF S_D.
5. The distance between the supporting plane of the door back face and the nearest point on the tool contact surface is equal to a predefined value τ_c, i.e.
$$\min_{P \in C_S} {}^D p_z = \tau_c, \tag{3.8}$$
where ${}^D p_z$ is the third component of the vector p with respect to RF S_D.

The distances τ_c and τ_{eL} are used to take into account the inaccuracy of the measurement of the door position in relation to the robot.

Furthermore, it is required that two consecutive configurations q_k and q_{k-1} must be sufficiently close to each other in both the task space and the joint space. In particular, they must fulfill the following *smoothness conditions*:

1. The Chebyshev distance between q_k and q_{k-1} in the joint space is smaller than a threshold value τ_q.
2. Let ${}^D T_G(q_k)$ be the pose of the robot tool with respect to S_D, which is achieved with the joint configuration q_k. The poses ${}^D T_G(q_k)$ and ${}^D T_G(q_{k-1})$ must be sufficiently

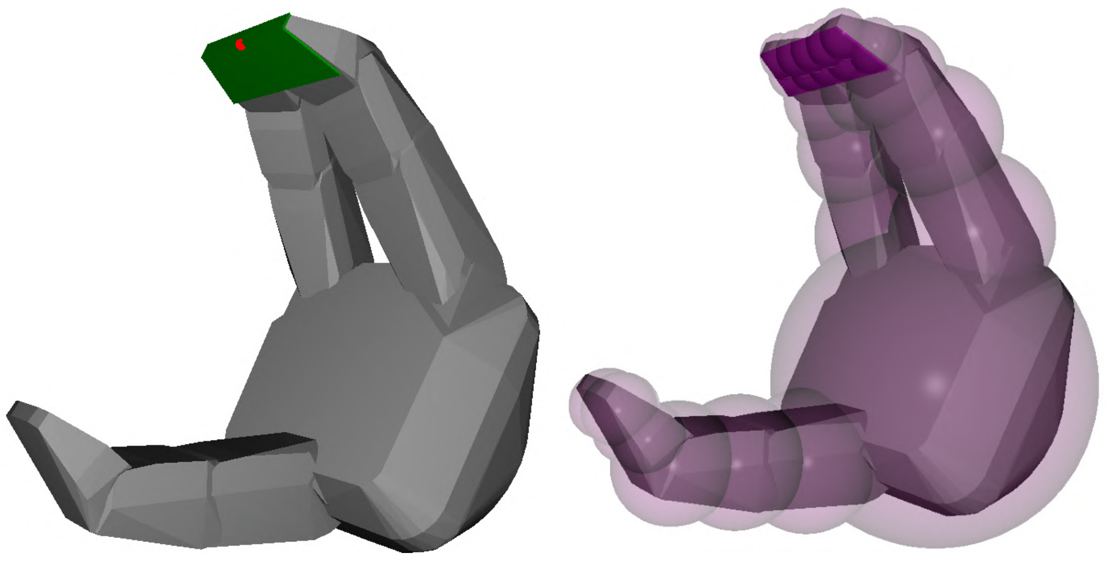
Figure 3.3: Collision model of the Robotiq 3-Finger Gripper, with the tool contact surface marked in green (left) and an approximation with a set of spheres (right). The reference tool contact point is represented by a red circle.

similar. Two poses T and T' are considered to be similar if

$$\delta_R\left(T, T'\right) \leq \varepsilon_R \tag{3.9}$$

and

$$\|t - t'\| \leq \varepsilon_t, \tag{3.10}$$

where t and t' are the translation vectors of T and T', and ε_R and ε_t are user-defined threshold values. The function δ_R calculates the angular distance along the shortest path between two orientations:

$$\delta_R(T, T') = \arccos\left(\frac{\mathrm{tr}(R^T R') - 1}{2}\right), \tag{3.11}$$

where R and R' are the rotation matrices of T and T'.

The quality of a path Q is measured according to two requirements. The first requirement is that the change of the joint variables between two consecutive points of the path is as small as possible. In addition, it is desirable that the contact point between the gripper and the door panel is not very close to the edges of the door panel, as the contact may be lost due to inaccuracies in the tool positioning and the estimated position of the door. These two requirements are mathematically encoded in the following cost function

$$E\left(Q\right) = \sum_{k=1}^{n-1} \delta_q\left(q_k, q_{k-1}\right) + \gamma \sum_{k=0}^{n-1} \beta\left(q_k\right), \tag{3.12}$$

where $\delta_q\left(q, q'\right)$ denotes the sum of the squared differences between the corresponding elements of the two configurations q and q', $\beta\left(q\right)$ is a cost that penalizes contacts near the edge of the door panel and γ is a weighting factor. Note that δ_q accounts for the shortest angular distance, wrapping the resulting differences into $[-\pi, \pi]$. The cost $\beta\left(q\right)$ is defined by

$$\beta\left(q\right) = \max\left\{0,\ \tau_{eH} - \min\left\{{}^{D}p_{c,x}\left(q\right), {}^{D}p_{c,y}\left(q\right)\right\}\right\}, \tag{3.13}$$

where ${}^{D}p_c\left(q\right)$ is the position of the reference tool contact point with respect to S_D when the robot is in configuration q and τ_{eH} is a threshold value defined by the user. If the weighting factor γ is set to a large value, the algorithm generates paths in which the reference tool contact point is positioned at a distance of at least τ_{eH} from the nearest edge of the door panel in all states in which this is possible.

If there are multiple paths that meet the above conditions, the one that minimizes the cost (3.12) is preferred.

3.3 Method

3.3.1 Planning Door-Opening Paths

In this section, an algorithm for planning door-opening paths is proposed that solves the problem defined in Section 3.2. The algorithm is formally presented as Algorithm 4. The inputs to the algorithm are the door model $\mathcal{D}$ described in Section 3.2.1, the environment model $\mathcal{E}$, the robot model $\mathcal{R}$ and the robot pose with respect to the world RF represented by ${}^{W}T_R$. The environment model and the robot model are used for collision detection. The proposed algorithm can be used in combination with an arbitrary environment and robot models that are suitable for collision detection. The models used in the experiments reported in this chapter are briefly described in Section 3.3.3.

The core idea of the proposed algorithm is to construct a feasible door-opening path by exploring different feasible contact point combinations and selecting the one that minimizes the cost (3.12).

The algorithm begins by calculating a set C of poses of the robot tool with respect to S_D that fulfill the state conditions 3, 4 and 5 specified in Section 3.2.2. These poses are referred to as *contact poses* in this chapter. A point on the tool contact surface is selected as the *reference tool contact point*. It should be noted that the tool does not necessarily touch the door panel at the reference tool contact point. This point is only used to define the distribution of the contact poses on the door panel.

The set of contact poses C is computed by uniformly sampling the back face of the door panel, and each sample point P is used to define a sample position of the tool. A set of random tool orientations is then selected for each sample point P. For a given sample point P and a given tool orientation, the position of the tool is calculated such that P defines the x- and y-coordinates of the reference tool contact point with respect to S_D, while its z-coordinate is such that the distance between the door back face and the closest point on the tool contact surface is equal to τ_c. State condition 5 is therefore fulfilled. Each tool pose computed in this way is checked for collision with the door panel. If the tool does not collide with the door panel (state condition 3) and the orthogonal projection of the tool contact surface overlaps sufficiently with the back face (state condition 4), the computed pose is stored in C. Set C is referred to in this chapter as the *contact set*.

From the set C, a *contact pose graph* Γ is constructed, where the nodes represent the tool poses from C. Since the nodes of the graph Γ represent tool poses with respect to S_D, for simplicity, the nodes of Γ are referred to as poses in this chapter and are represented by homogeneous transformation matrices ${}^{D}T_{G_i}$, where G denotes the robot tool RF S_G and i stands for the node index. In addition, the symbol C is used to denote the set of all nodes of the graph Γ. Two poses ${}^{D}T_{G_i}$ and ${}^{D}T_{G_j}$ are connected in Γ if they fulfill the second smoothness condition specified in Section 3.2.2.

The proposed algorithm creates a set of feasible configurations F_k for each state k, connects feasible configurations to paths Q and selects the optimal path that minimizes the cost (3.12) as the final result.

Algorithm 4 Multi-contact path planning for door opening

```
1: procedure PATH($\mathcal{D}, \mathcal{E}, \mathcal{R}, {}^{W}T_R$)
2:     Create a contact set $C$ from $l$, $\mu$ and $\mathcal{R}$.
3:     Create graph $\Gamma$ from $C$.
4:     for $k \leftarrow 0$ to $n-1$ do
5:         Compute ${}^{W}T_{D(k)}$ by Eqs. (3.2)–(3.3).
6:         $F_k \leftarrow \emptyset$
7:         for each ${}^{D}T_{G_i} \in C$ do
8:             Compute ${}^{R}T_{G_i(k)}$ by Eq. (3.14).
9:             $F_{ik} \leftarrow FeasibleConfigs\left({}^{R}T_{G_i(k)}, \mathcal{E}, \mathcal{R}\right)$
10:            $F_k \leftarrow F_k \cup F_{ik}$
11:        end for
12:        for each $q_{k,j} \in F_k$ do
13:            if $k = 0$ then
14:                $E_{k,j} \leftarrow \gamma\beta\left(q_{k,j}\right)$
15:                $Q_{k,j} \leftarrow \left(q_{k,j}\right)$
16:            else
17:                $U \leftarrow Neighbors\left(q_{k,j}, F_{k-1}\right)$
18:                if $U$ is empty then
19:                    Remove $q_{k,j}$ from $F_k$.
20:                else
21:                    for each $q_{k-1,u} \in U$ do
22:                        $\Delta E_u \leftarrow \delta_q\left(q_{k,j}, q_{k-1,u}\right) + E_{k-1,u}$
23:                    end for
24:                    $u^* \leftarrow \arg\min_{u \in U} \Delta E_u$
25:                    $E_{k,j} \leftarrow \Delta E_{u^*} + \gamma\beta\left(q_{k,j}\right)$
26:                    $Q_{k,j} \leftarrow \left(Q_{k-1,u^*}, q_{k,j}\right)$
27:                end if
28:            end if
29:        end for
30:    end for
31:    $j^* \leftarrow \arg\min_{j \in F_{n-1}} E_{n-1,j}$
32:    return $Q_{n-1,j^*}$
33: end procedure
```

The set F_k is formed from the set C by considering each contact pose ${}^{D}T_{G_i} \in C$ and determining all feasible configurations that bring the tool into this pose in state k. For a contact pose ${}^{D}T_{G_i}$, the corresponding tool pose with respect to the robot base reference frame S_R in state k can be computed by

$$ {}^{R}T_{G_i(k)} = {}^{W}T_R^{-1}\,{}^{W}T_{D(k)}\,{}^{D}T_{G_i}, \tag{3.14}$$

where ${}^{W}T_{D(k)}$ denotes the pose of S_D with respect to S_W in state k, computed according to Equation (3.3), and ${}^{W}T_R$ is the pose of S_R in relation to the world RF S_W, which is assumed known.

The function *FeasibleConfigs* computes the inverse kinematics to find joint configurations that reach the given tool pose ${}^{R}T_G$, checks if the configurations obtained as solutions of the inverse kinematics are reachable considering the kinematic constraints and the joint limits of the robot, and checks if there is a collision between the robot and the environment. The function returns a set of feasible configurations F_{ik}, which is then added to the set F_k. In some cases, F_{ik} can be an empty set. Each configuration $q_{k,j}$ is associated with a path $Q_{k,j}$ with the cost $E_{k,j}$ calculated according to Equation (3.12).

Paths Q are obtained by recursively connecting configurations from each set F_k with configurations from the set F_{k-1}, where $k = 1, \dots, n-1$. For a given configuration $q_{k,j}$, the function *Neighbors* in step 17 returns a set U of feasible configurations $q_{k-1,u}$ from the set F_{k-1}, that satisfy the smoothness conditions specified in Section 3.2.2. This operation is facilitated by adjacency relations in the contact graph Γ. Each configuration $q_{k,j}$ is derived from a contact pose ${}^{D}T_{G_i} \in C$ by calculating ${}^{R}T_{G_i(k)}$ according to Equation (3.14) and calculating the inverse kinematics for this tool pose, and each contact pose is represented by a node of the contact graph Γ. The node of the contact graph Γ, which represents the contact pose from which a configuration $q_{k,j}$ is derived, is referred to in this chapter as the corresponding node of $q_{k,j}$. It is assumed that the algorithm tracks the relationships between configurations $q_{k,j}$ and their corresponding nodes in Γ. For a given configuration $q_{k,j}$, the function *Neighbors* only examines configurations $q_{k-1,u}$ from the set F_{k-1} that have the same corresponding node in the contact graph Γ as $q_{k,j}$ or whose corresponding node is adjacent to the corresponding node of $q_{k,j}$. This enables an efficient search for configurations in F_{k-1} to which $q_{k,j}$ can be connected by restricting the search to a small number of configurations that fulfill the second smoothness condition.

The configuration $q_{k,j}$ is appended to the path with minimum cost that starts from the zero-state (steps 21 – 26). If $q_{k,j}$ cannot be connected to a configuration in F_{k-1}, it is removed from F_k (steps 18 – 19). In this way, there is a feasible path $Q_{k,j}$ for each configuration in F_k that starts from the zero-state.

The result of the described path planning process can be represented by an exploration graph whose nodes are feasible configurations contained in the sets F_k, where $k = 0, \dots, n-1$. Each node from a set F_k, $k = 1, \dots, n-1$ is connected to a node in the set F_{k-1}. An example of an exploration graph created with the proposed algorithm is shown in Figure 3.4. For better visibility, the algorithm is only calculated for 8 states and the number of contact poses from which the feasible configurations are calculated is limited to 50.

The last step of the algorithm is to find the path $Q_{n-1,j}$ for which the cost (3.12) is minimal, among the paths associated with the last state configurations $q_{n-1,j} \in F_{n-1}$. This path is the final result of the proposed algorithm.

3.3.2 Pre-Computation of the Contact Pose Graph

To achieve high efficiency, data that does not depend on the position, size and state of the door is pre-computed and used in real-time path planning. This approach leverages the fact that the door panel has a simple shape that differs from case to case only by its size. This enables pre-computation of the contact set C in advance and creating the graph Γ from the set C, store it and use it in real time. In order for the algorithm to be applied to open doors of different sizes, the contact set and the corresponding graph for the maximum expected door size are computed and adapted to an actual door size, which may vary from case to case. Let $l_{y,max}$ and $l_{z,max}$ be the maximum expected width and height of a door panel and let l_y and l_z be its actual width and height respectively. Given a contact set C_{max} of the largest expected door panel, C is computed as the subset of C_{max}, which consists of all poses of C_{max} for which the x- and y-coordinates of the reference tool contact point with respect to S_D are smaller than l_y and l_z, respectively. The regions of the xy-plane of S_D corresponding to C_{max} and C are shown in Figure 3.5.

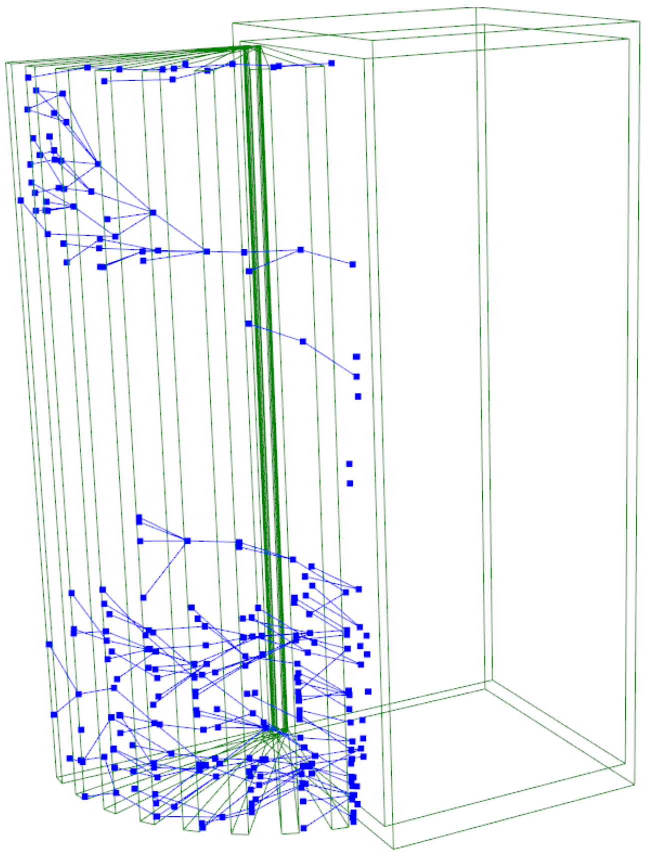

Figure 3.4: Exploration graph created by the proposed algorithm.

Let Γ_{max} be the graph computed for the contact set C_{max}. For a set C corresponding to an actual door panel, the graph Γ is extracted from Γ_{max} by extracting the nodes corresponding to the poses from C. Note that extracting of Γ from a pre-computed Γ_{max} is computationally much less expensive than generating C and creating Γ from it, which involves collision detection and adjusting the distance of the tool from the back face of the door panel for all poses in C as well as searching for neighborhood relations between all nodes in Γ.

3.3.3 Collision Detection

Since set C contains only tool poses in which the tool is not in collision with the door panel, the collision between the tool and the door panel does not need to be checked when executing Algorithm 4 if C is pre-computed and loaded at the beginning of the algorithm execution. However, the collision with the static part of the piece of furniture must still be checked. A contact pose in set C represents a gripper pose with respect to the door panel. Since the pose of the door panel with respect to the static part of the furniture is different for different states k, the same contact pose in C corresponds to different gripper poses with respect to the static part of the furniture. Therefore, when checking the feasibility of a particular configuration, the collision with the static part of the furniture must be checked for each state k.

Furthermore, since a pose ${}^{D}T_{G_i} \in C$ corresponds to different poses ${}^{R}T_{G_i(k)}$ in different states k, the feasibility of a configuration in terms of robot kinematics and joint constraints as well as the collision between the robot arm and both the static part of the piece of furniture and the door panel must be checked for each state k.

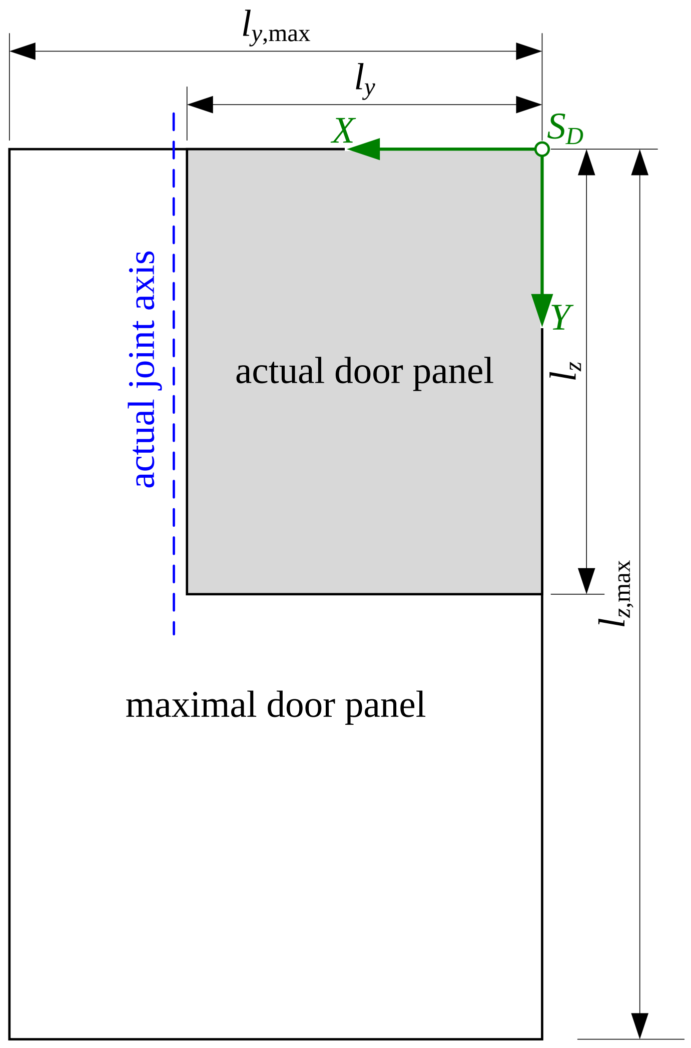

Figure 3.5: Maximal expected and actual door panel.

Regarding the robot arm, it was determined that for this door-opening task, only the forearm can be in collision with the cabinet. Therefore, the collision model was simplified by approximating the forearm using spherocylinders (see Figure 3.6). This significantly improves computational efficiency while maintaining sufficient accuracy for collision detection between the robot and the environment.

Since the proposed approach allows for any collision checking method, two methods have been implemented: an SDF-based collision detection and a mesh-based collision detection using the Flexible Collision Library (FCL) (Pan, Chitta and Manocha, 2012).

For the SDF-based collision checking, the gripper is approximated by a set of spheres such that their union completely contains the gripper, as shown in Figure 3.3. Each sphere is defined by a vector ${}^{G}c$, which represents the position of the sphere center with respect to the gripper RF S_G and the sphere radius r_c. This model allows an efficient collision check under the assumption that the signed distance function $f(p)$ of the environment is available, i.e., if the distance of any point p in the robot's workspace to the nearest obstacle is known.

For a specific gripper pose in relation to the robot base RF ${}^{R}T_G$ and the robot pose with respect to the world RF ${}^{W}T_R$, the position of the center of each sphere with respect to S_W can be computed using

$$ {}^{W}c = {}^{W}T_R \, {}^{R}T_G \, {}^{G}c. \tag{3.15}$$

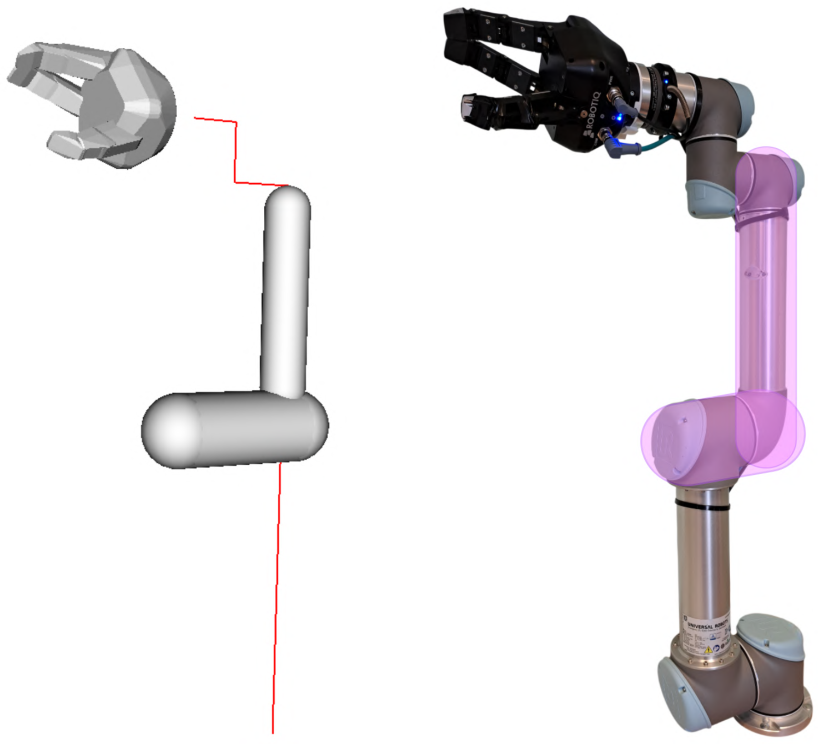

Figure 3.6: Illustration of the UR5 forearm modeled with spherocylinders for efficient collision detection. The 3D visualization on the left shows the spherocylinders and the gripper mesh, with the remaining links represented as red lines. The image on the right displays the physical UR5 robot with the spherocylinders overlaid on the corresponding links.

If

$$f\left({}^{W}c\right) > r_c \tag{3.16}$$

applies to all spheres, then the gripper is not in collision with the environment. In the experiments reported in this chapter, the robot environment consists of two objects: a cabinet with a door and a floor surface. The robot itself is represented by spheres for the gripper and two cylinders for the robot's third link, since the third link is the part most likely to come into contact with the cabinet. The VolumeNet model (Cupec and Đurović, 2018) is used to represent the cabinet, including the door panel, and the floor surface. This model provides a conservative approximation of the signed distance function for a given point in 3D space.

The second implemented method for collision detection is the classic algorithm from the Flexible Collision Library (FCL). Only the discrete collision detection method is used, without proximity query and without continuous collision detection. For this method, the gripper, the robot links and the environment were represented by 3D meshes. To create a safety margin for collision checks, an inflated collision mesh of the Robotiq 3-Finger Gripper model[1] was used. Each part of the articulated mesh model was inflated by 10% and the number of their faces was reduced by a factor of 5, resulting in 1098 faces for the gripper. This allowed a significant reduction in the computational cost of the collision check. If the number of faces was reduced even further, a risk of losing the desired shape of the collision model would arise, compromising the accuracy of collision

[1]3D models of the gripper were sourced from the Robotiq GitHub repository: https://github.com/ros-industrial-attic/robotiq.

detection. The differences between the original collision model and the inflated mesh are illustrated in Figure 3.7.

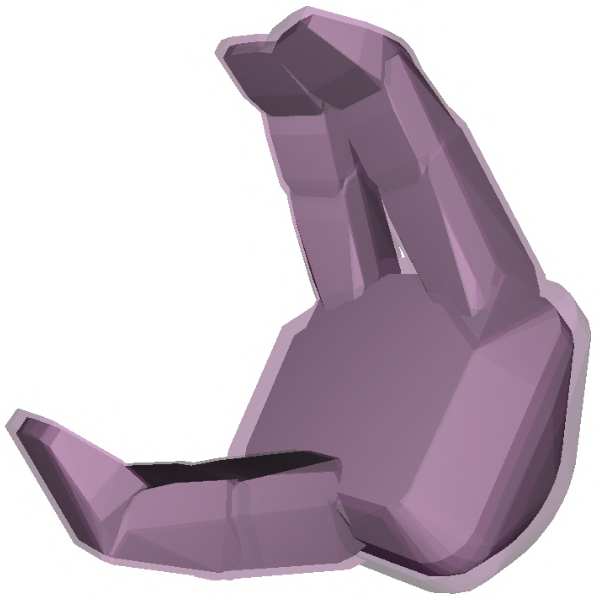

Figure 3.7: Visualization of the original and inflated collision mesh models of the Robotiq 3-Finger Gripper. The inflated model is shown semi-transparent to highlight the added safety margin around the original model.

Although the gripper is assumed to be rigid, the Robotiq 3-Finger Gripper has some flexibility in the fingers. To reduce the possibility of unintended movement, a small stabilizing plate was added to the fingers, highlighted in purple in Figure 3.3.

3.3.4 Node Sampling

Algorithm 4 is intended to explain the basic idea of the proposed multi-contact path planning approach in a clear and simple way. In this section, it is explained how steps 7 – 10 of this algorithm can be implemented efficiently. This efficient implementation is used in the experiments described in Section 3.5.

In Algorithm 4, the feasibility of all solutions of the inverse kinematics is checked for all contact poses in C in order to form the set F_k. A higher efficiency can be achieved if F_k is formed by considering only the contact poses for which the inverse kinematics allows solutions that can be connected to configurations in F_{k-1}, according to the smoothness conditions mentioned in Section 3.3.1. To achieve this, the contact graph Γ can be used. For each state k, a set $H_k \subseteq C$ is first formed, which consists of the contact poses corresponding to all configurations in the set F_{k-1}. For this purpose, the relationships between the configurations and their corresponding contact poses are recorded, as already mentioned in Section 3.3.1. This set is extended by adding all contact poses that are adjacent to its elements in the contact graph Γ, resulting in the set $H_k^+ \subseteq C$. The set F_k is formed by considering only the contact poses from the set H_k^+ instead of the entire set C. This strategy reduces computation time significantly, as the set H_k^+ is usually much smaller than C. This is particularly noticeable at the beginning of the door-opening path, when the door is only slightly open. The narrow gap significantly reduces the number of feasible initial poses, making the set H_0 significantly smaller than C. Because of this, the size of H_1^+ is also limited, as it is constructed from the poses from H_0 and their neighboring poses. This reduction propagates through the subsequent states, restricting not only the set F_1 but also the sizes of the sets F_k in later states. Furthermore, as the door-opening motion continues, kinematic constraints and joint limits may filter out poses in F_{k-1} that become infeasible in later states, reducing the expansion of the following sets F_k.

The speed of the algorithm can be further increased by limiting the size of the set F_k using the following procedure. First, a contact pose is drawn at random from H_k^+. Then, a set of feasible configurations is derived from this contact pose using the $FeasibleConfigs$ function and these configurations are added to the set F_k. This process ends when a predefined maximum number of samples $n_{s,max}$ is reached or the entire H_k^+ is exhausted.

The described sampling strategy is formally presented in Algorithm 5. The procedure is intended to replace steps 6 – 10 in Algorithm 4, optimizing the construction of the set F_k by restricting the contact pose search space and limiting the maximum number of valid configurations computed.

Algorithm 5 Sampling feasible configurations with neighborhood search

```
1:  procedure SampleFeasibleConfigs($k, F_{k-1}, C, \Gamma, n_{s,max}, \mathcal{E}, \mathcal{R}$)
2:      $H_k \leftarrow \emptyset$
3:      if $k = 0$ then
4:          $H_k^+ \leftarrow C$
5:      else
6:          for each configuration $q_{k-1,j} \in F_{k-1}$ do
7:              Retrieve contact pose $v \in C$ associated with $q_{k-1,j}$
8:              $H_k \leftarrow H_k \cup \{v\}$
9:          end for
10:         $H_k^+ \leftarrow H_k$
11:         for each $v \in H_k$ do
12:             Retrieve set of neighbors $N(v)$ of $v$ from $\Gamma$
13:             $H_k^+ \leftarrow H_k^+ \cup N(v)$
14:         end for
15:     end if
16:     $H_k^+ \leftarrow \text{Permute}(H_k^+)$
17:     $F_k \leftarrow \emptyset$
18:     while $|F_k| < n_{s,max}$ do
19:         ${}^D T_{G_i} \leftarrow$ next element of $H_k^+$
20:         Compute ${}^R T_{G_i(k)}$ by Eq. (3.14)
21:         $F_{ik} \leftarrow FeasibleConfigs\left({}^R T_{G_i(k)}, \mathcal{E}, \mathcal{R}\right)$
22:         $F_k \leftarrow F_k \cup F_{ik}$
23:     end while
24:     return $F_k$
25: end procedure
```

3.3.5 Computational Complexity

Assuming that the contact set C and the contact pose graph Γ are pre-computed, the critical operation with respect to the computational complexity of the proposed algorithm is the computation of the joint configurations from the contact poses and their validation. This process includes the calculation of the inverse kinematics and the collision check, the latter being much more computationally intensive. In the zero-state, the contact poses are taken from the set C and validated until the maximum number of samples $n_{s,max}$ is reached or the entire set C is exhausted. In cases where the piece of furniture is positioned with respect to the robot in such a way that only a few collision-free contact poses can be achieved, taking into account the kinematic constraints and joint limits of the robot, the algorithm must validate many contact poses before it reaches the maximum sampling limit $n_{s,max}$. On the other hand, in the best case, if all contact poses result in one or more feasible configurations, $n_{s,max}$ feasible configurations can be found after less than $n_{s,max}$ contact poses have been validated. The worst case is when the entire set C is examined without finding $n_{s,max}$ feasible configurations. Since the size of the contact set C is likely to be much larger than $n_{s,max}$, the number of validations, which can be regarded as a measure of computational complexity, for the zero-state can vary from below $n_{s,max}$ to $|C|$ depending on the pose of the furniture. The size of the contact set C is approximately proportional to the area of the door back face. For example, in the experiments reported in Section 3.5, the size of C is typically a few tens of thousands, depending on the door size, while the optimal value of $n_{s,max}$ in the experiments is 100. In cases where no feasible configuration for the zero-state is found, the computational cost of processing the zero-state is estimated to be $O(|C|)$.

After generating a set of feasible configurations for the zero-state F_0, the algorithm continues with the generation of sets F_k for the remaining states $k = 1, \ldots, n-1$. Since this procedure only considers the adjacent contact poses to those that led to feasible configurations in the previous state, the percentage of positive validations for the remaining states is very high. Therefore, the maximum sample limit $n_{s,max}$ is already reached after the validation of a relatively small number of contact poses. Consequently, the computational complexity for all states except the zero-state is estimated to be $O(n \cdot n_{s,max})$.

3.4 Experimental Setup

In order to study the benefits of multi-contact path planning, four distinct experiments were carried out: three quantitative evaluations conducted in a simulation environment and one qualitative experiment in a real-world setting. Each experiment involved the utilization of a Universal Robots UR5 robot manipulator, for which the analytic inverse kinematics were computed based on Keating and Cowan, 2016. The setup also included a Robotiq FT300-S force-torque sensor, alongside a Robotiq 3-Finger Gripper configured in "pinch mode." This mode entails a specific configuration of the gripper's fingers in which the two fingers on one side come together to grasp an object in a pinching motion, similar to how human fingers pinch.

3.4.1 Simulation Experiments

For the simulation experiments, the Robot Operating System (ROS) and the Gazebo simulator were utilized to place a cabinet inside an environment with the initial door state φ_0, as explained in Section 3.2.1, and perform a door-opening action with the UR5 robot. The RF of the robot base S_R aligns with the RF S_W, with a predetermined offset of 0.005 m along the z-direction. This adjustment prevents the collision between the robot and the floor in the simulation.

A total of 1000 cabinets with varying dimensions and poses were randomly generated in the simulator to evaluate which cabinet doors the robot was able to open. To distinguish successful from unsuccessful attempts, several conditions were defined, all of which had to be simultaneously satisfied. The first condition required that a complete trajectory for opening the door was successfully found, meaning the entire path necessary to open the door had to be successfully computed. The remaining conditions were evaluated only if this criterion was met. The second condition involved the execution of the planned trajectory and was satisfied if the robot reached the final waypoint. The third condition required that no collisions occurred between the robot and the static parts of the cabinet during execution. Finally, the fourth condition assessed whether the cabinet door was opened successfully, defined as reaching an opening angle of 90°, within a tolerance of $\pm 5^\circ$.

3.4.2 Generating Cabinets in Simulation

Table 3.1 shows the properties of the randomly generated cabinets in the simulation experiments. The width and height correspond to the dimensions of the cabinet door, while the position and orientation correspond to the axis of rotation of the door in the simulation world frame S_W. The z-coordinate of the frame S_A is calculated so that the cabinet stands on the floor, with the same offset along the z-direction as the robot base.

Table 3.1 Cabinet properties used in simulation experiments

	Property	Min. Value	Max. Value
Size	Width [m]	0.2	0.6
	Height [m]	0.2	0.8
Pose	Position in X [m]	-0.75	0.75
	Position in Y [m]	0.0	0.75
	Rotation [°]	-180	180

The space of the cabinet poses is defined by several criteria. The first criterion defines the position of the frame S_A in the xy-plane of S_W. The distance from S_A to S_W must be $> 0.3\,\mathrm{m}$ to avoid collisions of the robot's second link with any part of the cabinet. The second criterion constrains the allowable positions of the door panel throughout the door-opening motion within the xy-plane of the world frame S_W. In particular, for each of the 50 discrete door states where the door-opening angle φ is between 0° and 90°, the entire edge of the door panel, defined as the line segment connecting S_A to the outermost point of the panel S_D, must remain completely outside a circle with a radius of 0.3 m, the center of which is the origin of S_W. At the same time, the position of S_D must lie within a circular area with a radius of 0.9 m, the center of which is the origin of S_W. In this way it is ensured that the door remains free from obstruction by the base of the robot during the opening action, while remaining within the reach of the gripper. To ensure that the cabinet door is oriented towards the robot, the following must be satisfied:

$$ {}^{W}t_D \cdot {}^{W}z_D > 0, \tag{3.17} $$

where ${}^{W}t_D$ is the translation vector of ${}^{W}T_D$ as defined in Equation (3.3), and ${}^{W}z_D$ denotes the unit vector representing the z-axis of S_D relative to S_W at the zero-state $\varphi = 0$.

The illustration of the cabinet pose space is shown in Figure 3.8. This space of the cabinet poses was defined to ensure that the contact point of the gripper with the door lies in the robot's workspace. Note that although the gripper may reach a contact point, this does not mean that it can necessarily be oriented at that contact point so that it can push the door. Additionally, the space of feasible gripper poses within the robot's workspace is constrained by the obstacles present in the simulation environment. In this case, the obstacle refers to the static part of the cabinet.

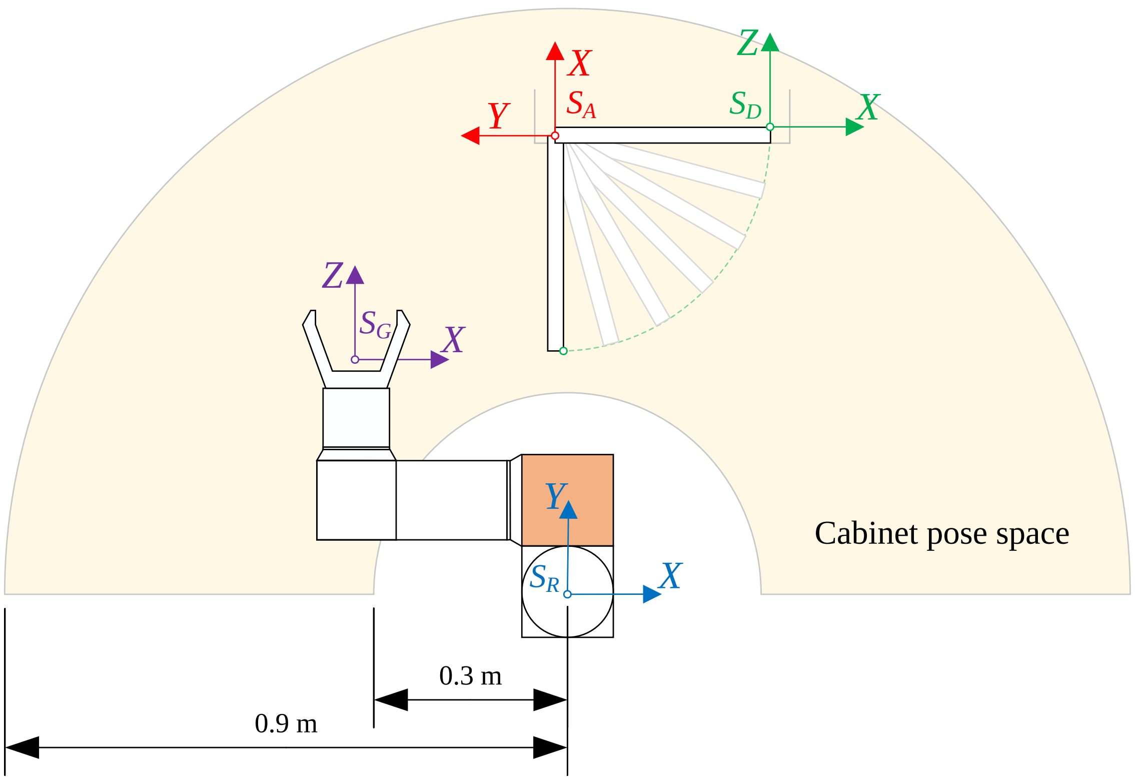

Figure 3.8: An illustration showing the pose space of the cabinet in the robot's environment. The figure illustrates the possible positions of the RF S_D during the entire door-opening movement. At all times, the origin of S_D must remain within the radius of 0.9 m. In addition, the line segment that connects S_A with the point S_D must remain outside the circular exclusion zone with a radius of 0.3 m. The defined cabinet pose space prevents contact between the generated cabinet and the second link of the robot (highlighted in orange) and ensures that the gripper can reach any potential contact point on the back of the door panel.

When the cabinet is created and placed in the simulation world, it is placed with its initial door state:

$$\varphi_0 = \mu \arcsin(\frac{\lambda + \frac{l_z}{2}}{l_y}), \tag{3.18}$$

where μ, l_y and l_z are the door parameters previously explained in Section 3.2.1, while λ represents the length of the push-latch mechanism. In the experiments reported in this chapter, $\lambda = 0.046$ m, a value commonly found in push-latch mechanisms. The geometric configuration defining the initial door state is depicted in Figure 3.9.

3.4.3 Generating Cabinet Poses in the Real-World Experiments

In the real-world experiments, a cabinet was randomly placed on the same table on which the robot was mounted. The cabinet poses were generated randomly within a constrained range, as summarized in Table 3.2. Since the robot was mounted at the center of the table with respect to its width, the positions of the cabinet were generated in such a way that all four corners of the cabinet's bottom plate must lie on the table, i.e.

$$|{}^{W}p_{x,i}| < \frac{w}{2}, \quad i = 1, \ldots, 4, \tag{3.19}$$

where ${}^{W}p_{x,i}$ denotes the x-coordinate of the i-th corner point on the bottom plate of the cabinet, represented in the world RF S_W, and w represents the table width. Notably, a

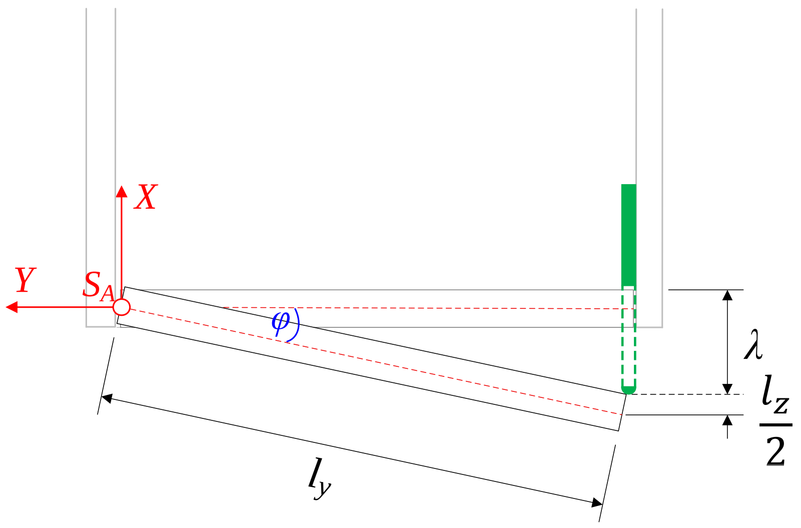

Figure 3.9: Illustration of the initial door state. The figure shows the cabinet door in its zero-state and an initial slightly open configuration caused by the extension of the push-latch mechanism. The push-latch mechanism, highlighted in green, has an extension length λ, while the dimensions of the door are defined by l_y and l_z.

similar check was not made for the position in y-direction, as it was already limited by the maximum value specified in Table 3.2. The defined cabinet pose space used in the real-world experiment is illustrated in Figure 3.10.

Table 3.2 Range of cabinet poses in real-world experiments

Property	**Min. Value**	**Max. Value**
Position in X [m]	-0.36	0.0
Position in Y [m]	0.41	0.71
Rotation [°]	-20	20

3.5 Results

This section presents the results of simulation and real-world experiments conducted to evaluate the proposed approach for door-opening tasks. The evaluation includes both single-contact and multi-contact scenarios, with additional comparisons to a baseline method based on Task Space Regions (TSR) (Berenson, Srinivasa and Kuffner, 2011). For clarity, the methods in this section are labeled as follows: the single-contact method is denoted *SC*, the multi-contact method that uses SDF for collision checking is denoted *MC-SDF*, the multi-contact approach that uses FCL for collision checking is denoted *MC-FCL*, and the hybrid method that begins the opening motion using the handle and transitions to multi-contact with SDF-based collision checking is denoted as *HMC-SDF*. An instance of each of the aforementioned methods is shown in Figure 3.13. Table 3.6 shows the results of all methods tested in the simulation. While these results provide insight into how a particular instance passed or failed in the simulation and reflect the overall results of all simulation experiments, it is also important to examine why certain instances were unsuccessful.

The analysis of unsuccessful attempts in the simulation was done using a distribution of failed instances for each method, categorizing these instances into four primary failure types: contact dynamics issues, missed approach paths, rotational collisions, and panel

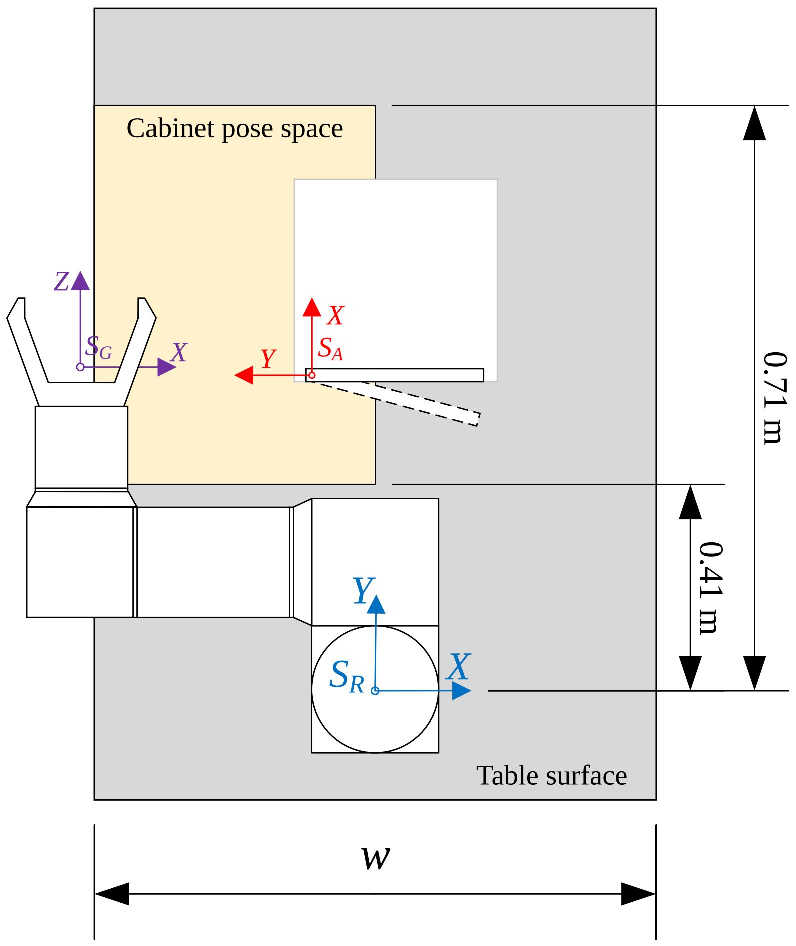

Figure 3.10: The image showcases the defined cabinet pose space in the real-world experiments. The gray area represents the table surface, on which the robot is mounted. Within this space, the pose of reference frame S_A is randomly generated, confined to the yellow region.

collisions. Contact dynamics issues refer to errors in contact handling in the Gazebo simulator, particularly near the door edges, where the gripper either slips through or loses contact unexpectedly. Cases of missed approach occurred when the gripper fails to reach the back face of the door panel and instead pushes the door shut during the approach. Rotational collisions occurred when the gripper contacts the static part of the cabinet while transitioning between planned contact points, often early in the trajectory. The last type of error observed was the collision with the panel, where parts of the robot other than the gripper come into contact with the door panel, disrupting the motion.

Figure 3.11 illustrates the main failure types observed during the simulation experiments. In panel (a), the gripper establishes contact near the edge of the door, but due to unstable or inaccurate contact handling in the Gazebo simulator, the door either closes unexpectedly or the gripper slips through without opening it. Panel (b) shows a missed approach where an incorrectly calculated contact point causes the gripper to collide with the door panel when trying to insert it into the gap between the door and the static part of the cabinet. This often results in the door closing partially or completely. As the robot does not receive any feedback on the pose of the door during execution, it is unable to adjust the motion and try again. In panel (c), the robot reaches the initial contact point but collides with the static part of the cabinet while transitioning to the next point of the trajectory. This is recorded as a failure, regardless of whether the door is ultimately opened. Finally, panel (d) shows a collision with the panel, where the robot starts to open the door but makes unintended contact with the door panel with a body part other than the gripper, as shown in light red in the figure.

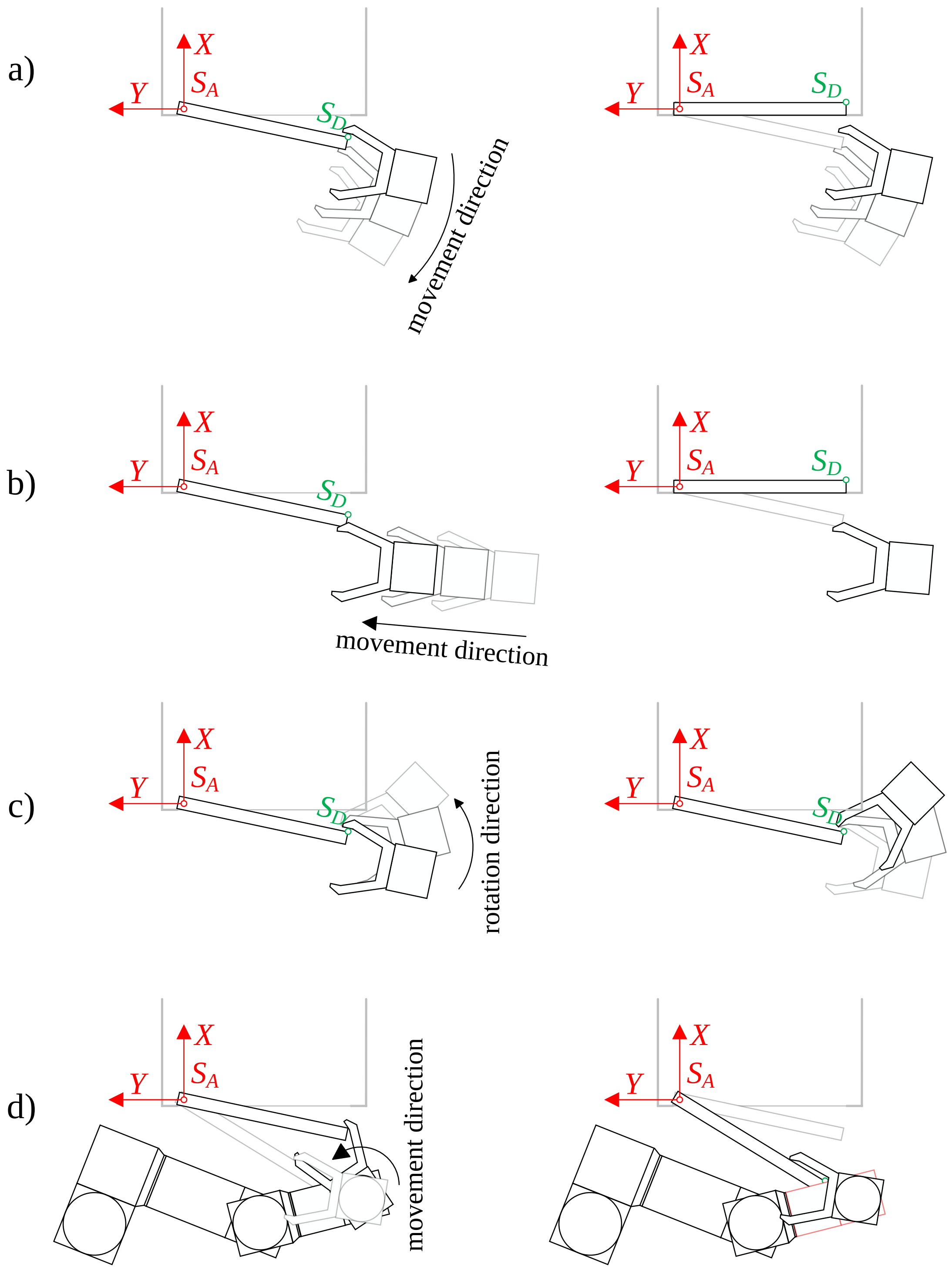

Figure 3.11: Illustration of failure types observed in the simulation experiments. The failures are grouped into four categories: (a) contact dynamics issues, (b) missed approach paths, (c) rotational collisions and (d) panel collisions (highlighted in red). For each category, the corresponding planned motion and the resulting failure outcome are shown in adjacent columns.

Furthermore, an analysis of parameter selection is provided by testing two parameters: the maximum number of samples $n_{s,max}$ and the number of door states n, along with an execution time analysis to compare the computational performance of the proposed method with the TSR approach.

3.5.1 TSR-Based Single-Contact Method

To create a basis for comparison, the Task Space Regions framework was utilized to perform a door-opening task on cabinets with handles.

The generated cabinets were identical to those described in Section 3.4.2, with the handles positioned in the vertical center on the side opposite the door joint axis. Since the handle provided a fixed contact point, the push-latch mechanism was unnecessary and the cabinets were placed in the zero-state in the simulation.

Task Space Regions (TSR) Berenson, Srinivasa and Kuffner, 2011 is a sampling-based framework for manipulation planning that incorporates constraints on the end-effector's pose into the motion optimization problem. In addition to defining constraints, the TSR framework also integrates constraint satisfaction strategies with a general planning algorithm, making it suitable for whole-body manipulation planning.

A TSR representation describes end-effector constraints as subsets of *SE(3)*. In this representation, one or more constraints, called Task Space Regions (TSR), can be defined. Each TSR consists of three main components:

- ${}^{0}T_{w}$ - a transformation from the world RF to the TSR RF S_w,
- ${}^{w}T_{e}$ - an offset transformation of the end-effector,
- ${}^{w}B$ - a 6×2 matrix specifying end-effector constraints in RF S_w.

The bounds matrix ${}^{w}B^{T}$ is defined as:

$$ {}^{w}B^{T} = \begin{bmatrix} x_{\min} & y_{\min} & z_{\min} & \psi_{\min} & \theta_{\min} & \phi_{\min} \\ x_{\max} & y_{\max} & z_{\max} & \psi_{\max} & \theta_{\max} & \phi_{\max} \end{bmatrix}, \tag{3.20} $$

where the first three columns define position constraints, while the last three columns define rotation constraints using the roll-pitch-yaw angle convention. The TSR framework also supports TSR chains, i.e. sequences of linked TSRs that define more complex constraints for manipulation planning. Each TSR chain consists of one or more TSRs that define the respective constraint problem. Each chain can be differentiated according to whether it is used for sampling goals, constraining poses along a trajectory, or both.

The implementation of the TSR method was based on the publicly available code provided by the authors (Berenson, Srinivasa and Kuffner, 2011). The TSR constraints utilized in this chapter correspond to the approach described in Arduengo, Torras and Sentis, 2021.

The path planning process consists of two phases. In the first phase, the robot moves from its home position to the contact pose, while in the second phase, it maintains contact with the handle and moves until the door reaches the fully open state. The goal of the first phase is a single point that can be reached with multiple inverse kinematics solutions, while in the second phase, the objective is to successfully execute the door-opening motion until the door is fully open. To achieve this, three TSR chains were used, each consisting of a single TSR constraint. The parameter p_{sample}, which controls the probability of goal sampling in the planning process, was set to 0.1, according to the value proposed in Berenson, Srinivasa and Kuffner, 2011.

The first TSR chain was defined as a goal sampling chain, specifying a fixed gripper pose at a single point of contact. The purpose of this TSR was to allow multiple inverse kinematics solutions for that specific pose. This chain is formally defined as:

$$ {}^{0}T_{w} = {}^{W}T_{A}, \tag{3.21} $$

$$ {}^{w}T_{e} = {}^{A}T_{E}, \tag{3.22} $$

$$ {}^{w}B = [\mathbf{0}]_{6\times 2}, \tag{3.23} $$

where S_E is the flange RF of the robot. The second TSR chain constrains the trajectory and shares ${}^{0}T_{w}$ and ${}^{w}T_{e}$ with the first TSR chain. The boundary matrix is defined as follows:

$$ {}^{w}B^{T} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \phi_{\min} \\ 0 & 0 & 0 & 0 & 0 & \phi_{\max} \end{bmatrix}, \tag{3.24} $$

where the bounds $\phi_{\min}$ and $\phi_{\max}$ depend on the axis-side parameter μ as follows:

$$ \phi_{\min}, \phi_{\max} = \begin{cases} \left(-\frac{\pi}{2}, 0\right), & \text{if } \mu = -1 \\ \left(0, \frac{\pi}{2}\right), & \text{if } \mu = 1 \end{cases} $$

The last TSR chain is another goal-sampling chain that shares ${}^{w}T_{e}$ and ${}^{w}B$ with the first TSR chain. Its ${}^{0}T_{w}$ is defined as:

$$ {}^{0}T_{w} = {}^{W}T_{A}{}^{A}T_{A'}\left(\mu\frac{\pi}{2}\right). \tag{3.25} $$

The results of the TSR method are summarized in Table 3.6. In 693 instances, the TSR method did not find a viable way to open the doors by pulling on the handle. Of the 307 instances where a path was found, the method successfully opened a cabinet in 298 cases, giving a success rate of 97.1% when a path was found. Even though the results from Table 3.6 show that in 6 cases there was a collision with the static part of the cabinet, in all 9 cases the robot positioned itself too close to the door so that its body collided with the door panel. In the 6 cases mentioned, this collision caused the cabinet to break in the simulation which resulted in the robot colliding with the static part.

Since the TSR method was evaluated on the cabinets with handles, there was only a single point of contact on the door and a single gripper pose that was feasible throughout the door-opening motion. This significantly reduced the solution space for the door-opening task. An example of door opening using the TSR method is shown in Figure 3.13. Although the cabinet features a handle, the gripper does not physically grasp it in the simulation. Instead, the interaction is simulated by virtually attaching the gripper to the door in the Gazebo simulator. This approach avoids closing the kinematic loop and helps prevent potential issues with contact dynamics in Gazebo, while still enabling realistic execution of the opening motion.

3.5.2 Single-Contact Simulation Experiment

In addition to the comparison with a TSR-based single-contact planning algorithm, the proposed algorithm was also compared with a single-contact planning method for opening doors with a push-latch mechanism by pressing the back face of the door panel. In this single-contact path planning approach, a contact point is selected on the back face of the door panel and a path is planned so that the position and orientation of the tool is constant in relation to the door panel.

First, a set of feasible configurations F_0 is determined using the method described in Section 3.3.1, a configuration from this set is randomly selected and the starting pose ${}^{D}T_{G(0)}$ is computed from this configuration. Then, for each door state $k = 0, \ldots, n-1$, ${}^{R}T_{G(k)}$ is computed such that the relative orientation between the gripper and the door remains constant:

$$^{D}R_{G(k)} = {}^{D}R_{G(0)}. \tag{3.26}$$

This ensures that the gripper maintains contact at a fixed point on the door while preserving a constant orientation throughout the door-opening motion. An example of the single-contact (SC) door-opening method executed in simulation is shown in Figure 3.13.

Table 3.6 summarizes the results of the SC experiment. No path was found in 546 cases. Of the remaining 454 cases, 427 cases were successful, resulting in a success rate of 94.05% for the cases in which a path was found. In 26 instances, the robot failed to open the doors within the allowed error margin. In the vast majority of these cases, 22 out of 26, this happened due to inaccurate contact handling in the Gazebo simulator near the edges of the cabinet door model. These issues occurred when the robot could only pull the door by the edge, as the gripper was unable to reach further into the back surface due to its size. In one of these instances, the cabinet broke and the robot collided with the static part. In addition, in 5 cases, the robot slightly missed the approach and closed the door while trying to reach the back surface of the panel. Another failure that did not result in the door opening fully occurred during the approach part of the path, where the gripper accidentally touched the door as it approached the contact point, causing the door to close. In one case, a collision with the static part of the cabinet caused the cabinet to break completely, which was due to unstable contact dynamics in the Gazebo simulator.

The critical part of the trajectory is the establishment of the first contact of the tool with the door, as the door is only slightly open in the initial state and the tool must be inserted into a narrow gap between the door and the static part of the furniture. Attempting to extend the safety boundary between the tool and the environment resulted in the algorithm failing to find a feasible contact pose in many cases where such a pose actually exists. However, when compared to the TSR method, the single-contact approach has the advantage that there are multiple possible contact points instead of just one. This expands the solution space for door opening, as the gripper can make contact at multiple points on the back face with multiple orientations.

3.5.3 Multi-Contact Parameter Selection

The proposed algorithm for multi-contact path planning has several user-defined parameters, which are described in Section 3.3. Two key parameters were tested: the maximum number of samples $n_{s,max}$ and the number of uniformly distributed door states n. These two parameters significantly affect the computational performance of the proposed algorithm, as explained in Section 3.3.5. Smaller values of these parameters lead to faster execution. On the other hand, if $n_{s,max}$ is too small, the algorithm may overlook some existing solutions. The other parameter, n, determines the sampling resolution of the resulting path. For smaller n, fewer points of the path are defined by the planning algorithm and more are generated by the interpolation between consecutive points by the robot control system. Since only the points generated by the planning algorithm are guaranteed to be collision-free, a sampling resolution that is too low increases the probability of collisions between the robot and the furniture parts. To determine a suitable value for $n_{s,max}$, a performance analysis was performed in which the number of states n was set to 40. The results are shown in Table 3.3. The results show that the method runs fastest when $n_{s,max} = 10$, with an average execution time of 0.02781 s per instance. However, increasing the value to 100 significantly increases the number of successfully generated paths, reaching a maximum of 976 out of 1000 attempts. A further increase in the value does not improve the number of generated paths, but increases the average runtime considerably. Due to the trade-off between the number of generated paths and the execution speed, $n_{s,max} = 100$ was chosen for the simulation experiments.

Table 3.3 Execution time analysis for different $n_{s,max}$ values ($N = 1000$ trials, $n = 40$)

$n_{s,max}$	**Min. [s]**	**Max. [s]**	**Avg. [s]**	**Paths Found**
10	0.01622	0.07653	0.02781	938
100	0.04875	0.19950	0.07975	976
1000	0.06852	0.54700	0.39760	976
10000	0.06890	4.15700	2.98500	976

Table 3.4 summarizes the results of the execution time analysis for the number of door states n, where the maximum number of samples $n_{s,max}$ was set to 100. The results show that the number of successfully found paths remains constant at 976 out of 1000 trials, regardless of the value of n. However, the number of successful experiments varies with n. Specifically, for $n = 30$, 787 instances were successful, while increasing n to 40 resulted in 958 successful instances. This indicates that n has no influence on whether a feasible path is found, but rather on how this path is executed. Even though $n = 30$ results in the fastest execution time, it can lead to collisions with the parts of the furniture. On the other hand, a higher n improves the resolution and improves the success rate of the path execution, but slightly increases the computing cost. In the simulation experiments, $n = 40$ is used as it provides a balance between the computational efficiency and the trajectory smoothness. The values of other user-defined parameters used in the simulations are listed in Table 3.5.

Table 3.4 Execution time analysis for different number of states n ($N = 1000$ trials, $n_{s,max} = 100$)

n	**Min. [s]**	**Max. [s]**	**Avg. [s]**	**Paths Found**
30	0.03844	0.1924	0.06979	976
40	0.04875	0.19950	0.07975	976
50	0.05257	0.2164	0.088	976

Table 3.5 Parameter values of the path planning algorithm used in the experiments

τ_c	τ_{eL}	τ_{eH}	τ_q	ϵ_R	γ
6 mm	7 mm	30 mm	45°	15°	10000
ϵ_t	$l_{y,max}$	$l_{z,max}$	$n_{s,max}$	n	
50 mm	0.6 m	1 m	100	40	

3.5.4 Multi-Contact Simulation Experiment

In the multi-contact experiment, the proposed approach was used to generate the multi-contact path to open the cabinet door. The approach was tested using the signed distance function (MC-SDF) and FCL (MC-FCL) to investigate how the choice of collision detection method influences the results. By comparing the results with the baseline TSR method described in Section 3.5.1 and the SC approach in Section 3.5.2, the advantages of the proposed method are highlighted, particularly its ability to expand the space of feasible door-opening paths.

The results of the multi-contact experiments are shown in Table 3.6. Compared to the single-contact approach and the TSR method, the MC approach was able to find significantly more paths. MC-SDF managed to find the path in 976 instances, while MC-FCL found the path in 977 instances. This shows that compared to the SC approach and the TSR method, the ability to change the contact point in the door-opening motion significantly increases the space of feasible paths. Based on the overall success rates of 95.8% for MC-SDF and 95.4% for MC-FCL, it can be seen that the methods perform similarly well. This shows the modularity of the proposed path-planning approach when it comes to collision detection.

However, in some cases the methods fail to open the doors successfully. In 4 cases, the gripper collides with the static part in the MC-SDF method, usually at the beginning of door opening due to a rotation of the gripper between two contact poses. This means that there was no collision between the robot and the static part at these contact points, but on the way between these two contact poses, the gripper nicked the static part, making the respective case unsuccessful. With MC-FCL, the gripper also touched the static part in 7 cases. Similar to the SC experiments, there were instances in which the robot executed the trajectory without colliding with the static part of the cabinet, but failed to open the doors. With the MC-SDF method, the robot mostly struggled with the approach path, which happened in 11 out of 976 cases where a path was found. In contrast, a similar error occurred 8 times with the MC-FCL method. The inability to open doors due to inaccurate contact handling occurred 3 times with the MC-SDF method and 8 times with the MC-FCL method. However, significantly fewer cases of this type occurred with the MC approach than with the SC approach. This was mainly due to the fact that the gripper was able to reach deeper into the door, where the likelihood of unstable contact was lower. There was also a single instance in MC-FCL where the robot touched the door with one of its links, preventing it from opening, an issue that did not occur with MC-SDF. The distribution of the failed instances for the MC approach is shown in Figure 3.12.

An example of the multi-contact door opening is shown in Figure 3.13. The robot approaches the door and makes the first contact on the edge of the door, shown in the second column of the figure. By adjusting the gripper pose in each state, the robot slides on the back surface of the door, opening it until reaching the end of the path.

3.5.5 Handle-to-Multi-Contact Method

In addition to the previously described single-contact and multi-contact strategies, a hybrid method, referred to as *HMC-SDF*, was evaluated on cabinets equipped with handles. This method was introduced to demonstrate that the proposed multi-contact approach remains effective even when a handle is present.

The robot first attempts to open the door by pulling on the handle until further motion becomes infeasible due to kinematic limitations or collision constraints. At this point, the door state is evaluated to determine whether it has been opened sufficiently, as defined by the condition:

$$\varphi_h \geq \varphi_0, \tag{3.27}$$

where φ_h denotes the door state after the handle-based motion, and φ_0 represents the initial door state defined in Equation (3.18). If this condition is satisfied, the gripper detaches from the handle and repositions to continue the opening motion using the MC-SDF method. Otherwise, the path is considered not found. This transition enables the robot to complete the door-opening task in scenarios where a purely handle-based method would otherwise fail.

The results of the HMC-SDF method are presented in Table 3.6. Compared to the TSR-based method, the HMC-SDF approach successfully found a significantly larger number of feasible paths, totaling 895 cases. This improvement arises from the fact that the opening motion is not limited to handle-based manipulation. When the handle-opening motion becomes infeasible, the method transitions to the multi-contact strategy, allowing the robot to continue opening the door, confirming the advantage of utilizing multiple contact points. The overall success rate of the HMC-SDF method is 99.33% for all cases where a path was found, representing the highest success rate among all tested methods. This performance can be attributed to the door being more opened during the initial handle-based manipulation compared to the doors with the push-latch mechanism, which enables the multi-contact approach to reach further inside the door panel from the start and reduces the likelihood of unreliable contact dynamic with the panel or the unwanted contact with the cabinet's static structure. However, the total number of successfully found paths remains lower than in the MC-SDF and MC-FCL methods, as the initial contact with the door is constrained to a single fixed point on the handle.

There were six instances in which the simulation failed. In one case, during the multi-contact stage, the robot slipped from the door due to issues in the contact dynamics, leaving the door only partially opened. In two cases, the robot successfully completed the handle-opening phase but, while repositioning for the multi-contact stage, accidentally bumped into the door, causing it to close slightly. In the remaining three cases, the gripper began opening the door using the multi-contact strategy but applied excessive force during the motion, resulting in the simulated door model breaking.

Table 3.6 Results of the TSR, single-contact, multi-contact with SDF collision checking, multi-contact with FCL, and handle-to-multi-contact methods in the simulation experiments ($N = 1000$ trials)

Path Found	Path Executed	Collision-Free	Door Opened	Number of Instances				
				TSR	**SC**	**MC-SDF**	**MC-FCL**	**HMC-SDF**
×	-	-	-	693	546	24	**23**	105
✓	×	×	×	4	0	0	0	1
✓	×	×	✓	1	0	0	0	0
✓	×	✓	×	0	0	0	0	0
✓	×	✓	✓	2	0	0	0	1
✓	✓	×	×	1	1	0	0	0
✓	✓	×	✓	1	0	4	7	1
✓	✓	✓	×	0	26	14	16	3
✓	✓	✓	✓	298	427	**958**	954	889

3.5.6 Real-World Experiments

In addition to the simulation experiments, a series of real-world experiments was conducted with a real robot to test whether the paths generated by the proposed algorithm would allow the robot to open doors. Three different cabinets were used in the experiments, as shown in Figure 3.14, with their dimensions listed in Table 3.7.

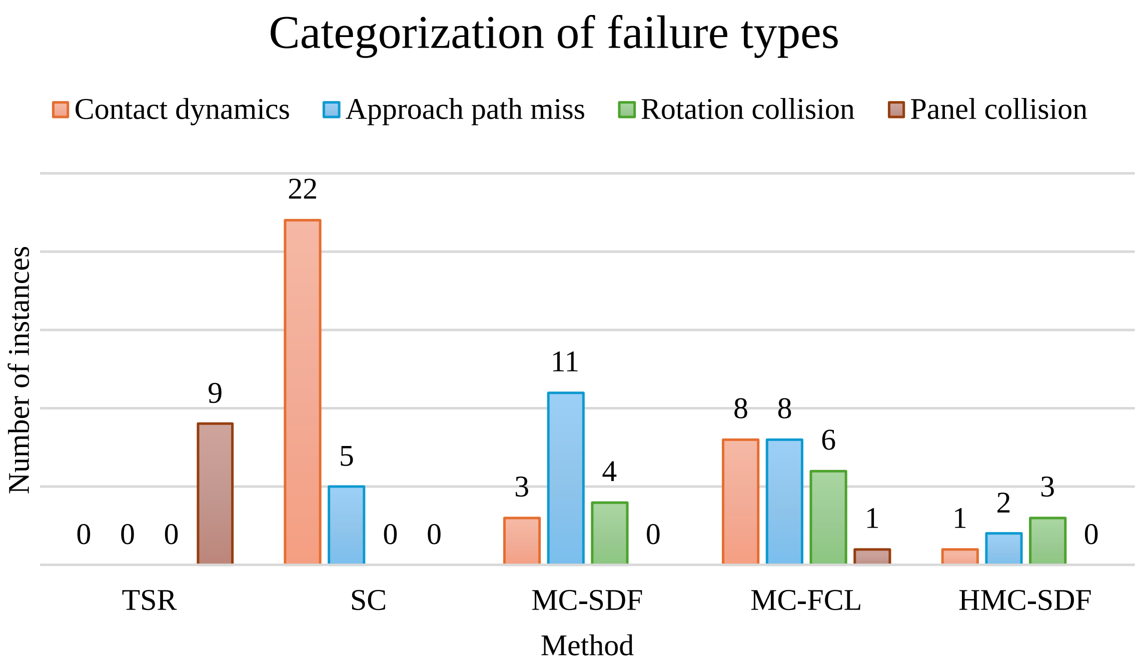

Figure 3.12: A graph showcasing the categorization of failed instances for each method in the simulation.

Table 3.7 Dimensions of cabinets used in real-world experiments

Cabinet	Width [m]	Height [m]	Thickness [m]
1	0.396	0.496	0.018
2	0.310	0.660	0.017
3	0.310	0.310	0.017

For each cabinet, poses were randomly generated, and the first 50 instances in which the MC-SDF method successfully found a feasible path and opened the cabinet in simulation were selected. These poses were then recreated using the corresponding real cabinet, and the UR5 robotic manipulator was used to perform the door-opening task for each pose. The results of the real-world experiments are summarized in Table 3.8.

Table 3.8 Summary of real-world experiment results ($N = 50$ trials per cabinet)

Cabinet	Failures	Successes	Success Rate [%]
1	3	47	94
2	1	49	98
3	0	50	100

For cabinet 1, 47 out of 50 executions were successful, resulting in a success rate of 94%. Three trials resulted in failure, each due to a different issue. In the first case, the robot executed the trajectory and attempted to move from one state to the next, turning the gripper so that it pushed on the door with excessive force. This triggered the safety threshold of the force-torque sensor, causing the robot to stop. In the second case, the robot initially made contact with the door, but slipped off during execution and continued to follow the planned trajectory without actually opening the door. In the third case, the robot missed the approach path and collided with the side of the door panel, which triggered the safety stop of the force-torque sensor.

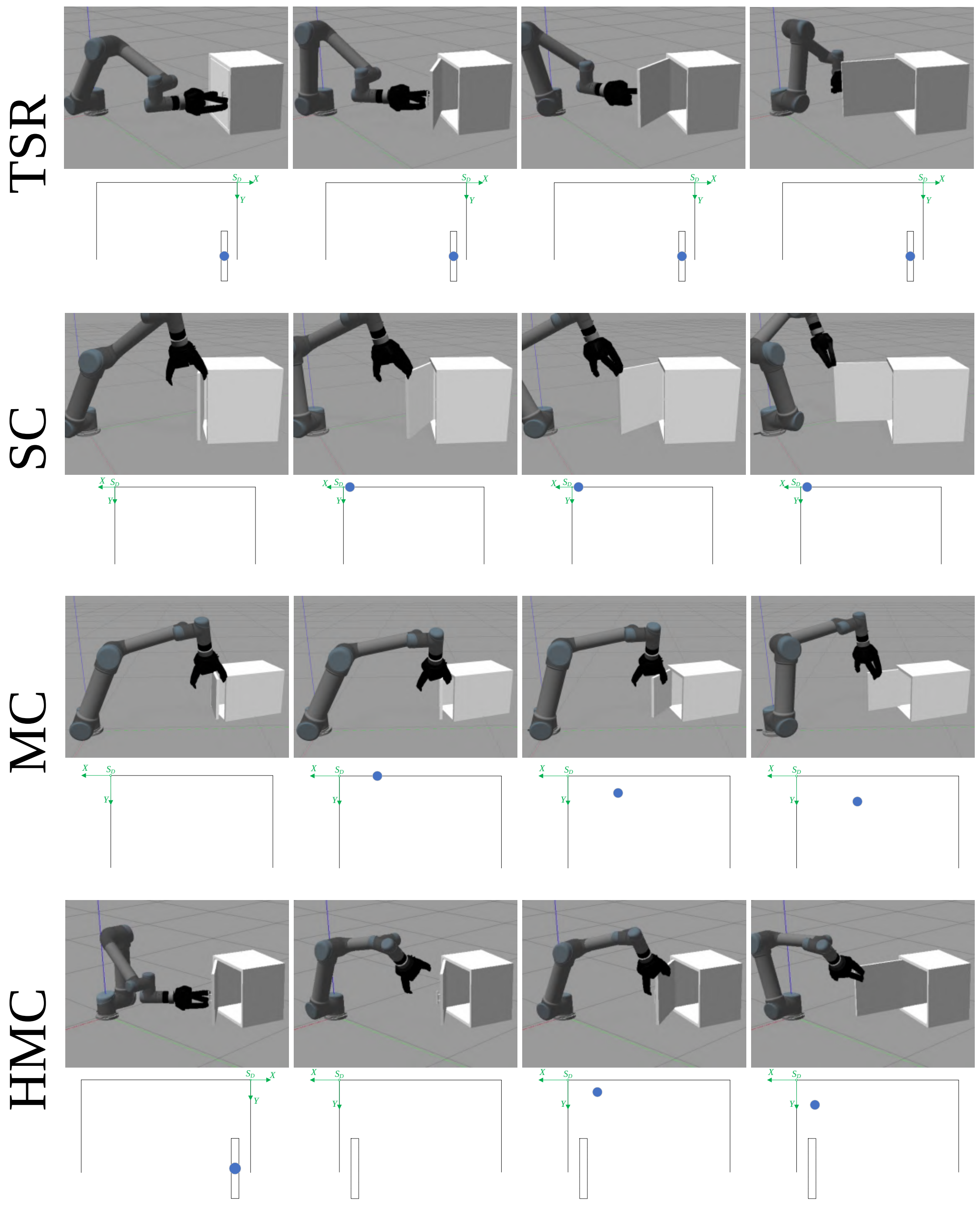

Figure 3.13: Examples of simulation experiments for each tested method. The top row shows the robot executing the door-opening task using the TSR, single-contact (SC), multi-contact (MC) and handle-to-multi-contact (HMC) approaches, respectively. The bottom row illustrates the corresponding contact locations on the door surface. The TSR method interacts with the door handle and maintains a single, fixed contact point. Similarly, the SC method maintains a consistent contact point on the back surface of the door throughout the motion. In contrast, the MC method changes the contact point during execution, enabling the gripper to slide along the door's back surface. The HMC method combines handle-based opening with the multi-contact approach. Contact points are shown as blue circles.

For cabinet 2, 49 out of 50 executions were successful, leading to a 98% success rate. The single failure occurred during the transition between contact points at the start of door opening, when the robot touched the static part of the cabinet.

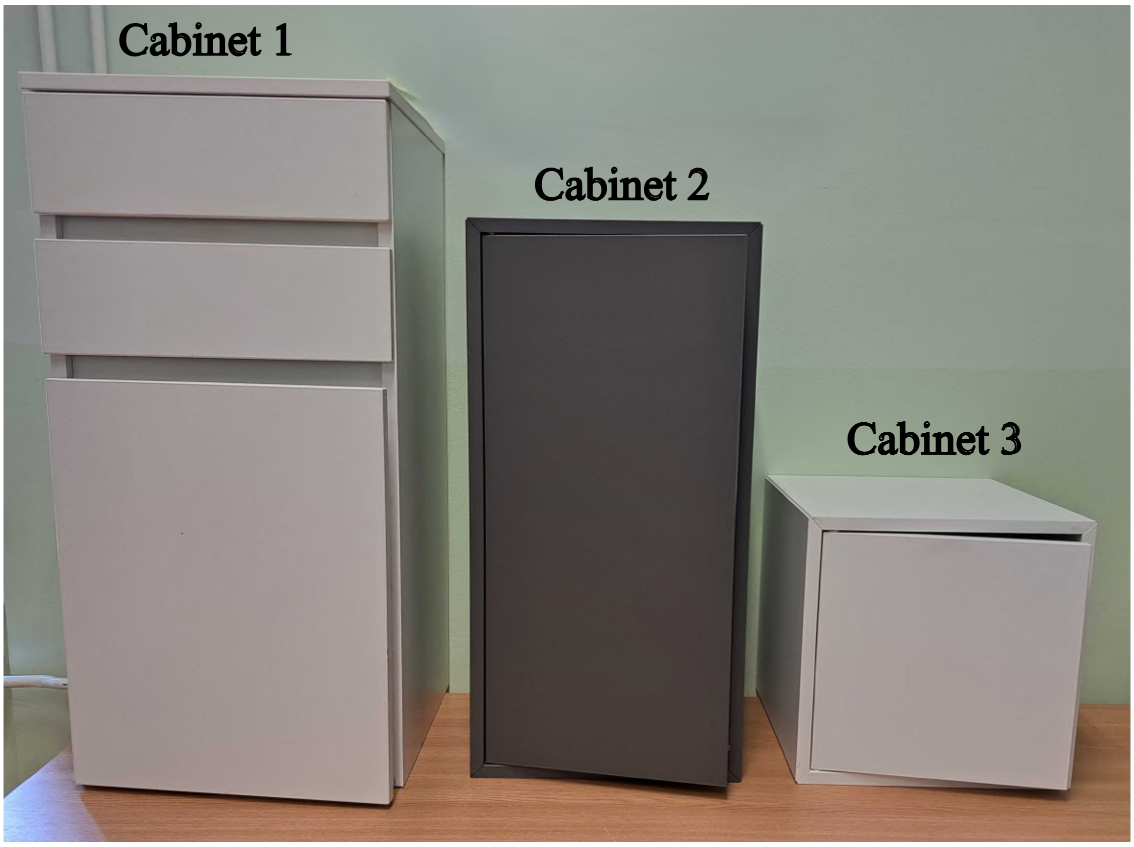

Figure 3.14: Cabinets used in the real-world experiments.

In the case of cabinet 3, all 50 trials were successful, resulting in a 100% success rate.

The observed failures were likely due to slight differences between the real and simulated cabinets, minor calibration errors in the tool used to position the cabinet, or slight inaccuracies in the cabinet placement. Despite the errors, overall success rates show that the proposed multi-contact approach can be effectively transferred from simulation to real-world execution across different cabinet types and configurations.

Figure 3.15 shows an example of a successful real-world experiment. In the first two subfigures, the robot is not in contact with the door, but is approaching the contact position on the door while avoiding collision with the static part of the cabinet. The gripper then passes through a series of points in space, sliding on the back surface of the door and opening it.

3.5.7 Execution Time Analysis

To evaluate the computational efficiency, a comparison of the execution time of three methods is performed: the TSR method, MC-SDF and MC-FCL. In addition, the MC-SDF method was used to demonstrate the efficiency of the precomputation of the contact pose graph. All experiments were run on an AMD Ryzen 9 7900 processor with 32 GB RAM in a Docker container with Ubuntu 20.04 and ROS Noetic.

The results of the method comparison are summarized in Table 3.9. The methods were compared in trials where a path was found. The results show that the MC-SDF method is the most efficient computationally, with an average execution time of 0.08064 s and it significantly outperforms the MC-FCL and the TSR methods. The maximum execution time of the TSR method indicates that certain planning instances require significantly more computation, likely due to the constraints imposed by task space sampling.

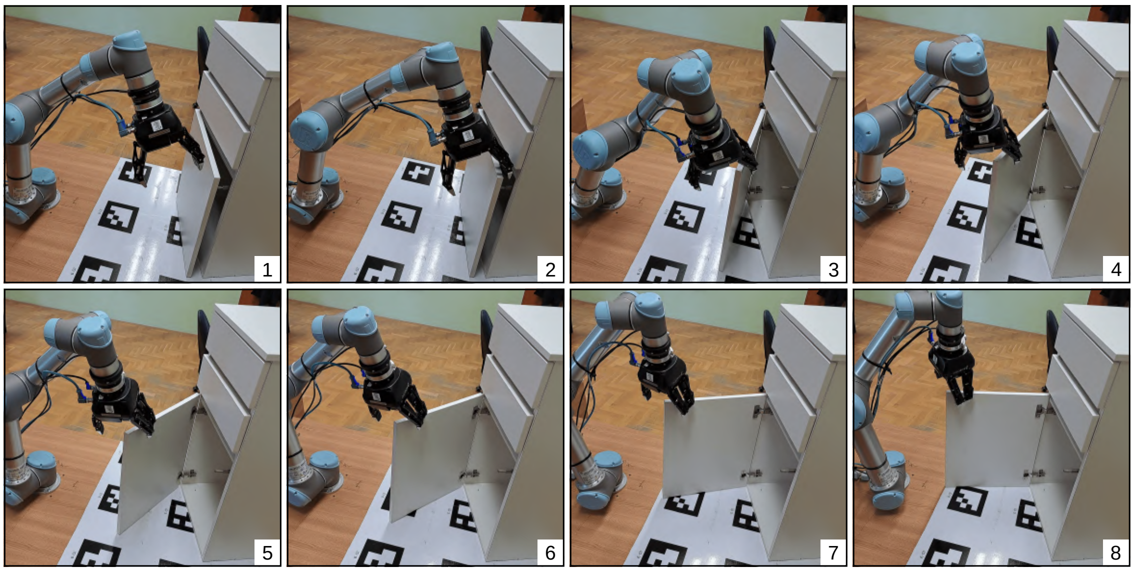

Figure 3.15: A sequence of images showing the multi-contact door opening in a real-world experiment. The images from 1 to 8 refer to the different states of door opening in a chronological order.

Because of the complex collision checking method in FCL, the MC-FCL method is significantly slower than the MC-SDF method, whose SDF-based collision checking method enables fast collision checking due to the simplicity of the gripper model approximation.

Table 3.9 Execution time comparison for different methods when a path is found ($N =$ 100 trials)

Method	**Min. [s]**	**Max. [s]**	**Avg. [s]**
TSR	0.0893	5.0015	0.9833
MC-SDF	**0.04884**	**0.1999**	**0.08064**
MC-FCL	0.2328	2.289	0.654

Table 3.10 shows the execution time of the MC-SDF multi-contact path planning, comparing scenarios where the contact pose graph is precomputed with scenarios where it is generated during the execution of the method. When a path was found, the algorithm ran on average 38 times faster when it loaded the precomputed contact pose graph from a file than when it generated it for each path planning instance. When no feasible path was found, the algorithm ran 23 times faster on average, showing considerable computational efficiency. However, the method was almost twice as slow when no path was found compared to when a path was generated. The results show that the execution time of the algorithm is significantly longer if no path is found. These results align with the theoretical analysis presented in Section 3.3.5 and confirm the expectations regarding the computational complexity in such cases.

3.6 Conclusion

This chapter proposed a novel path planning method for robotic door opening that considers multiple possible contact points between the robot end-effector and the door. The approach generates feasible and optimized trajectories for opening handleless doors while avoiding collisions and minimizing path length. The proposed method was published in Šimundić, Petrović and Cupec, 2026.

Table 3.10 Execution times of the MC-SDF method using online vs. pre-computed contact pose graphs ($N = 20$ trials)

Graph Creation	Outcome	Min. [s]	Max. [s]	Avg. [s]
Online Creation	Path Found	2.600	3.271	2.878
	Path Not Found	2.965	3.532	3.294
Pre-Computed	Path Found	0.050	0.154	0.075
	Path Not Found	0.065	0.193	0.141

Similar to the TSR method, the proposed approach is based on sampling tool poses and connecting the sampled poses to paths. However, instead of projecting the sampled poses onto a constraint manifold and checking for collisions with the door panel during planning, the proposed approach computes in advance a set of tool poses that are suitable for establishing the contact with the door surface and are collision-free with respect to the door panel. This set is used in online path planning, which significantly speeds up the path planning process. However, this strategy is only applicable to the specific problem of door opening, where the simple shape of the door panel allows the use of the same contact poses in different scenarios.

Although the proposed method may not be directly applicable to other robot manipulation tasks, tackling this problem efficiently is of great practical importance. Door opening is a fundamental skill for service robots operating in human-centered environments, as doors are ubiquitous in households, offices and various indoor spaces. A reliable and efficient solution to this task increases the autonomy and usability of a robot and is thus a crucial step towards fully functional assistive and service robotics.

To evaluate the proposed method, a systematic quantitative investigation was conducted through a series of simulations of cabinet door manipulation tasks, and experimental verification of the proposed method was provided in a real-world scenario featuring a door with a push-latch mechanism. The results demonstrate that the multi-contact approach significantly expands the solution space when compared to the single-contact path planning strategy which is dominantly used in the literature, thus outperforming state of the art in path planning for opening furniture doors.

The main problem in the practical application of the proposed method is the insertion of the robot tool into a narrow gap between the door panel and the static part of the furniture, which is opened by a push-latch mechanism. The success of this operation depends on very precise information about the position of the furniture and the door panel. An interesting avenue for future work would be to integrate the proposed method with a real-time perception system that could provide information about the pose, size and kinematic parameters of the target door. Another natural extension of this work, explored in the following chapter, is the incorporation of tactile and force sensing to detect collisions and correct the pose of the cabinet, thereby enabling reliable execution even in the presence of uncertainties in door pose and dimensions.

4 Visuo-Force-Tactile Integration for Door Opening

Robotic manipulation in unstructured environments remains one of the central challenges in modern robotics. Successful interaction with everyday objects requires not only precise perception and motion planning but also the ability to detect and recover from failures that arise during execution. Even small perception inaccuracies or calibration errors can cause differences between the perceived and actual geometry of the environment, leading to task failures, especially in manipulation tasks such as door opening.

Building upon the approaches introduced in the previous chapters, this chapter focuses on improving the robustness of door-opening tasks through the integration of multiple sensing modalities, particularly vision, force, and tactile data. The method presented in Chapter 2 enabled the detection and kinematic modeling of doors and drawers from visual data, providing data necessary for manipulation. Chapter 3 introduced a path planning strategy capable of generating feasible and efficient door-opening paths under ideal perception conditions. However, when transferring such planning methods to real-world scenarios, even small inaccuracies in the perceived object pose can lead to unstable contact interactions and failed manipulation attempts. These issues become particularly pronounced when the perception system is affected by calibration errors. These insights motivate the development of frameworks that use multimodal sensing to detect and recover from perception errors during door opening.

Recent research has explored recovery strategies for robotic manipulation using high-level learning paradigms such as large language models (LLMs) (Driess et al., 2023), visual language models (VLMs) (Lu et al., 2025), and reinforcement learning (RL) (Vats et al., 2025). These approaches have demonstrated promising results for recovery in manipulation tasks. However, these methods require a large amount of data that can be difficult to acquire in real-world settings. Other works propose recovery through visual feedback alone (Ebert et al., 2018; Byravan et al., 2017), but these approaches do not explicitly account for calibration inaccuracies and can struggle in conditions when the visibility is poor or the target object is occluded.

Existing door-opening frameworks (Wang et al., 2025; Vats, Likhachev and Kroemer, 2023; Kang et al., 2024b; Wei et al., 2025) are typically designed for doors equipped with handles. However, many modern furniture pieces lack handles, requiring precise end-effector positioning to establish effective contact. This increases the system's dependence on accurate perception and calibration, as even small pose estimation errors can lead to missed contacts or collisions.

In this chapter, a door-opening framework for opening handleless doors that integrates visual, force and tactile data for robust failure recovery is proposed. The system utilizes a 3D camera, a force-torque sensor, and a tactile sensor mounted on the robotic end-effector, which can be seen in Figure 4.1, to detect and correct errors in perception. The proposed system captures a sequence of RGB-D images of a cabinet door, reconstructs its 3D model, and uses this model to plan the corresponding door-opening motion. Errors in the reconstructed 3D model can lead to failure in the execution of the task. The hypothesis is that errors in the target object model stem mainly from three sources:

- Camera measurement noise – random, zero-mean noise with high spatial frequency;
- Systematic error due to inaccurate camera calibration;
- Segmentation error due to imperfect algorithms that assign image points to object surfaces.

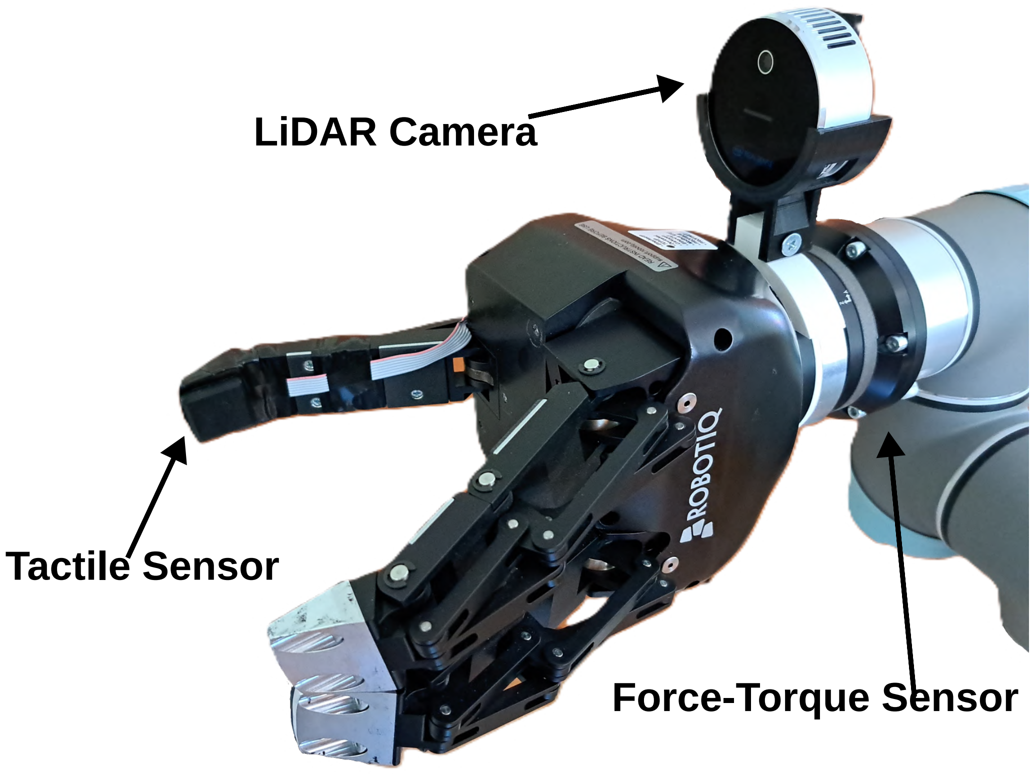

Figure 4.1: Overview of the equipment used in the proposed door-opening system, including a LiDAR camera, force-torque sensor, and tactile sensor mounted on the robotic end-effector.

Since camera measurement noise can be effectively averaged out when reconstructing surfaces from many image points, the dominant source of error is the systematic bias introduced by calibration inaccuracies. To address this problem, a method is proposed for correcting camera parameters to compensate for the systematic errors described earlier, based on the robot's interaction with the environment. In particular, the focus is placed on robot actions that result in a collision with the cabinet or a missed grasp. Such failures suggest that the current camera model is inaccurate, as the robot's motions are planned based on a cabinet model derived from perception. In response, the method corrects the parameters of the camera used for object detection and localization. Correcting the camera model leads to a more accurate cabinet model, which in turn allows the system to replan a more reliable door-opening trajectory.

4.1 Related Work

Robust door-opening manipulation relies not only on effective motion planning but also on accurate perception and adaptive control during interaction. Recent research has increasingly focused on integrating multiple sensing modalities, such as vision, force, and tactile feedback, to improve interaction with objects under different uncertainties, as well as on developing recovery mechanisms to handle execution failures. This section reviews related work in these two areas: visuo-force-tactile sensor fusion methods that combine information given by sensors for improved perception and manipulation, and failure recovery strategies that enable robots to adapt to unexpected events during door-opening and similar tasks.

4.1.1 Visuo-Force-Tactile Sensor Fusion

Some methods fuse vision and touch directly into shared representations for end-to-end learning. Huang et al., 2024 combine tactile sensing from custom fingertip arrays with multi-view RGB-D into a unified 3D model, from which imitation learning policies are trained to handle fragile objects and in-hand tools. Hansen et al., 2022 extend DrQv2 reinforcement learning with tactile gating and augmentation, ensuring policies learn to rely on tactile cues only when contact occurs. Similarly, Sferrazza et al., 2024 use a masked autoencoder to learn joint visual-tactile representations that generalize across manipulation tasks. Beyond touch, Li et al., 2022 show that adding audio alongside vision and touch in a self-attention policy improves precision in tasks like pouring and dense packing. These methods either demand large datasets (e.g., Huang et al., 2024), are limited to simulation due to data scarcity (Sferrazza et al., 2024), or face practical RL training challenges (Hansen et al., 2022). The method proposed in Li et al., 2022 even requires human demonstration for training.

Other work fuses modalities at the level of geometry and optimization. Caddeo et al., 2023 estimate in-hand 6D object pose from tactile images alone, using CNN-based contact point prediction refined through geometric constraints. Nonnengießer et al., 2025 merge point clouds from depth and tactile sensors, weighting contributions from each before applying a modified ICP algorithm to obtain the pose of an object. Li et al., 2025 refine visual pose estimates using a spring-mass optimization model grounded in touch and proprioception. The method checks the physical feasibility of the visual estimate and iteratively adjusts it, pulling poses toward tactile contacts while pushing them away from collisions, which improves robustness in real-world manipulation. While these methods can work well with small objects, it is uncertain how well they would work with larger objects such as doors and drawers.

Several approaches integrate vision with proprioceptive or force sensing for adaptive control. Kadalagere Sampath et al., 2025 combine YOLOv8-based (Jocher, Qiu and Chaurasia, 2023) object detection with force feedback in a dexterous hand, dynamically adjusting grasps via prediction models learned from demonstrations. Gupta et al., 2024 improve whole-body motion planning with proprioceptive corrections during grasping, while Wang et al., 2025 use visual inputs to structure a door-opening task into subtasks and proprioceptive forces to adapt execution in real time. Wei et al., 2025 take a generative route, where force feedback calibrates diffusion-based vision-guided trajectories to avoid unsafe contacts. As is often the case with deep learning-based methods, these methods can require large amounts of annotated data to be trained successfully.

Finally, visuo-tactile fusion has been explored for perception and reconstruction. Murali, Porr and Kaboli, 2025 combine point clouds from vision and tactile exploration to reconstruct occluded objects during decluttering. Fang et al., 2024 extend 3D Gaussian Splatting with tactile surface normals and vision-language guidance, sharpening scene geometry from sparse views.

While these methods achieve strong results in simulation and controlled real-world settings, they typically require extensive data or demonstrations, and their scalability to larger objects remains uncertain. Furthermore, they rely on precise camera calibration, as errors are assumed to arise from perception or policy learning, rather than from miscalibration itself.

4.1.2 Manipulation with Failure Recovery

Robust manipulation in unstructured environments requires effective failure recovery mechanisms to handle uncertainties and unexpected events during task execution. Ahmad et al., 2024 proposed behavior trees combined with motion generators (BTMG) for adaptable recovery behaviors, enabling robots to manage failures through hierarchical decision structures. Pan et al., 2022 presented a task and motion planning framework that explicitly considers failing executions, allowing robots to reason about potential failures during the planning phase. Ratner, Tomlin and Likhachev, 2023 integrated control-level discrepancy information directly into the planning process, enabling robots to operate with inaccurate models by adapting plans based on observed execution errors. These planning-based approaches provide systematic ways to handle failures but often require accurate failure detection and classification.

Several methods leverage contact and force information for failure recovery in manipulation tasks. Ren et al., 2025 developed collision-inclusive manipulation planning for occluded object grasping, utilizing compliant robot motions to handle unexpected contacts during manipulation. Jiang, Jia and Li, 2023 proposed contact-aware non-prehensile manipulation for object retrieval in cluttered environments, exploiting intentional contacts to achieve manipulation goals despite obstacles. Gavura et al., 2025 showed that haptic feedback-based robotic calibration improves sim-to-real transfer, enhancing manipulation robustness through better model alignment. Limoyo et al., 2018 developed self-calibration methods for mobile manipulators through contact-based interaction, correcting kinematic and sensor extrinsic parameters online. These contact-based methods improve robustness, but they rely on intentional contact, and have thus not been deployed for opening doors.

Learning-based approaches have shown promise for adaptive failure recovery. Ak, Aksoy and Sariel, 2023 proposed learning failure prevention skills for safe robot manipulation, training policies to anticipate and avoid common failure modes. Chen et al., 2024 automated robot failure recovery using vision-language models with optimized prompts, enabling robots to leverage semantic understanding for recovery strategy selection. However, these learning-based methods often require extensive training data and may not generalize well to novel failure scenarios.

System-level frameworks integrate multiple recovery strategies for comprehensive failure handling. Galbally et al., 2022 introduced Elly, a real-time failure recovery and data collection system that continuously monitors execution and triggers appropriate recovery behaviors. Burgess-Limerick, Leitner and Corke, 2023 enabled failure recovery for on-the-move mobile manipulation, addressing the unique challenges of recovering from failures during dynamic motion. Goller et al., 2025 developed fault handling mechanisms for model predictive interaction control, providing systematic approaches to detect and recover from various fault conditions. Despite these advances, existing methods often focus on specific failure types or require extensive prior knowledge about potential failure modes, limiting their applicability in unstructured environments.

4.2 Door-Opening Framework

This chapter proposes a robust and adaptable framework for opening handleless doors that integrates visual, force, and tactile sensing for decision-making in real-time. The

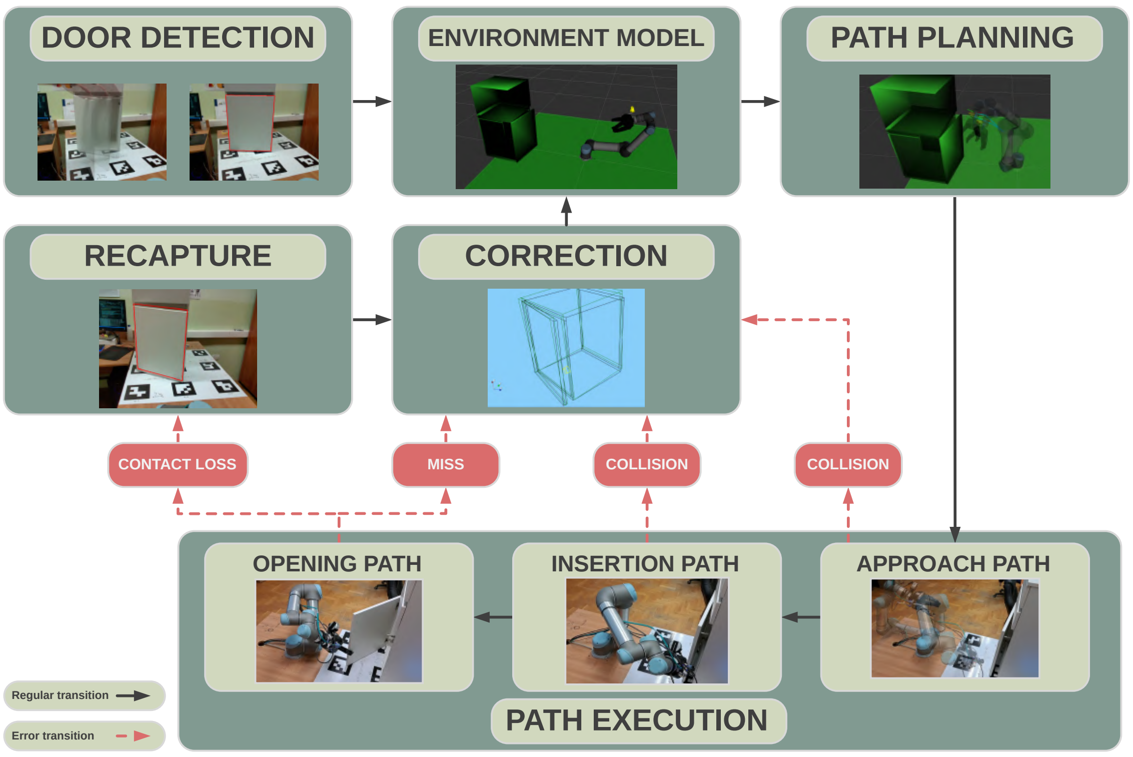

Figure 4.2: Overview of the proposed door-opening framework structured as a finite-state machine. Each state corresponds to a component responsible for a specific task, including door detection, planning, environment model creation and update, path execution, correction, and recapture. Transitions between the modules are triggered by visual, force or tactile feedback. Regular transitions (black) are used to control the nominal sequence, while error transitions (red) handle various failure cases, such as missed contacts, collisions or contact losses.

framework is structured as a finite-state machine, where each state corresponds to a specific phase of the door-opening task and transitions between them are triggered based on the sensory feedback. Two types of transitions are defined: regular transitions which occur when a task is completed successfully and error transitions which occur when failures are detected. The information obtained from the error transitions is used to correct the camera parameters and obtain a more accurate environment model based on these parameters, which is then used to plan the next trial. The hypothesis is that the information gathered by performing a series of tasks would gradually improve the robot's performance and reduce the error rate in future operation. An overview of the framework, together with its states and transitions is illustrated in Figure 4.2.

In this section, the sequence of decision-making steps that guide the robot through the door-opening process is outlined. The modular design of the framework enables that individual components or methods can be replaced, provided they produce the same output or serve an equivalent function.

4.2.1 Door Detection Module

The door-opening process begins with estimating a door model from visual data. To obtain an initial estimate of the door in the robot's workspace, the *Teaching by Human Demonstration* (DDD-THD) method from the *Door and Drawer Detector* (DDD) framework introduced in Chapter 2, is employed. The method estimates a door model, the complete description of which is provided in Section 4.4.1.

4.2.2 Correction Module

The door model estimated by the detection method mentioned in Section 4.2.1, is used to construct a 3D model of the cabinet, which serves as the environment model for path planning.

However, due to potential inaccuracies in the estimated parameters, the constructed environment model may deviate from the actual physical state of the cabinet, which can lead to manipulation failures. To address this issue, the system incorporates a *Correction Module*, which plays a critical role in enhancing robustness by compensating for perceptual errors and uncertainties in the environment. The correction procedure is invoked upon failure during execution, where it is triggered via an error transition, represented by red dashed lines in Figure 4.2. This failure may include a collision with the cabinet, a missed grasp or a contact loss. Once invoked, the correction procedure adjusts the estimated cabinet model based on the tool pose recorded at the moment of failure. The corrected model is then updated within the *Environment Model Module*.

By integrating the corrective mechanism, the system can adapt to discrepancies caused by inaccurate camera parameters, ensuring more reliable and consistent task execution. The details of the correction procedure are presented in Section 4.3.

4.2.3 Path Planning Module

The *Path Planning Module* generates feasible motion trajectories that enable the robot to approach, insert, and pull open the door using a predefined contact point on the back side of the door panel.

The output of this module is a set of candidate trajectories, each associated with a unique contact pose and compliant with kinematic constraints of the robot. One of these trajectories is selected for execution based on criteria described in Section 4.5.

4.2.4 Path Execution Module

The *Path Execution Module* is responsible for executing the trajectory selected during planning while continuously monitoring sensory feedback to detect and handle execution failures. The execution is divided into three phases: approach, insertion, and opening, each controlled by different monitoring criteria and state transitions.

During the approach path, the robot moves toward the cabinet. Throughout the motion, a force-torque sensor mounted on the robot wrist monitors external forces. If the measured force exceeds a predefined threshold, the robot stops, and the pose of the contact is recorded. This event triggers an error transition to the *Correction Module*, which corrects the camera parameters based on the contact and updates the environment model, enabling the Path Planning Module to replan accordingly.

If the approach completes successfully, the robot transitions to the insertion path, where the gripper moves into the gap between the cabinet and the door. Similarly to the approach path, any contact exceeding the force threshold leads to correction, environment update and replanning.

After a successful insertion, the robot proceeds with the door-opening motion. In this phase, tactile feedback from a sensor mounted on the gripper finger is used to confirm a contact with the door panel. If no significant tactile feedback is detected during the motion, the system assumes that the robot missed the contact and triggers an error transition to the Correction Module. Additionally, if contact is initially established but later lost during the door-opening motion, the system also registers this as a missed contact. In such cases, the system transitions to the Recapture Module to re-estimate the current door state.

4.2.5 Recapture Module

As mentioned in Section 4.2.4, the *Recapture Module* is activated when the tactile sensor loses contact during the door-opening process. Its primary function is to re-estimate the current state of the door, enabling the system to replan the door-opening trajectory accordingly. Such re-estimation is particularly important in scenarios where the door may have partially closed or returned to its initial state as a result of the forces of the hinge mechanism.

To achieve this, the *State Estimation* (DDD-SE) method from the *Door and Drawer Detector* (DDD) framework introduced in Chapter 2, is employed to estimate the current door configuration from a single RGB-D image.

4.3 Correction of Camera Parameters

In this section, a method that recalibrates camera parameters based on tool poses recorded in failure states is proposed.

4.3.1 Correction Model

This chapter considers a vision system that provides information about the robot's environment in the form of a set of polyhedra. More precisely, the surfaces of all objects in the robot's environment consist of polygonal planar faces f^e. The supporting plane of each face is defined by the equation:

$$n^T p_p = d, \tag{4.1}$$

where p_p is a point that lies on the plane, n is the unit vector that forms the normal to the plane, and d is the offset that represents the signed distance from the origin of a reference frame to the plane along the direction of the normal. This chapter considers an environment model A as consisting of a set of vertices, a set of planes, and associations between them.

It is assumed that the target object has been successfully recognized and modeled by a set of polyhedra, where each face of the model corresponds to a real face of the target object. However, due to inaccuracies in the vision system and camera calibration, the parameters of the faces, specifically the normals n and the offsets d of their supporting planes, may contain measurement errors. Furthermore, when an object face is represented in the image by thousands of points, the camera measurement noise is expected to cancel out by an appropriate plane estimation algorithm, such as least-squares fitting, and its influence on the estimated plane parameters could be neglected. The solution proposed in this chapter addresses inaccurate camera calibration as the main source of error in measurements obtained by vision.

An eye-in-hand configuration is employed, where an RGB-D camera is mounted on the robot's end-effector. When an RGB-D camera is employed to estimate the parameters of a target object for a door-opening task, its intrinsic parameters and its pose with respect to the robot's end-effector must be accurately determined through a suitable calibration process.

Let K be the camera intrinsic matrix defined by:

$$K = \begin{bmatrix} f_x & 0 & u_c \\ 0 & f_y & v_c \\ 0 & 0 & 1 \end{bmatrix}$$

and assume that parameters $f_x, f_y, u_c, v_c, \kappa$ and λ are obtained by an inaccurate calibration procedure and that they deviate from the true camera parameters. Let g_x, g_y, g_z, h_x, h_y and h_z define the correction parameters, which must be added to the camera parameters obtained by calibration to obtain the true camera parameters. The corrected camera intrinsic matrix is denoted by K', which is related to K by the following equation:

$$K' = K + L, \tag{4.2}$$

where

$$L = \begin{bmatrix} g_x & 0 & h_x \\ 0 & g_y & h_y \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.3}$$

Furthermore, the depth camera model utilized in this chapter is proposed in Khoshelham and Elberink, 2012. According to this model, the depth measurement z is computed from the disparity value δ using the following formula:

$$z = \frac{z_r}{1 + \frac{z_r}{\lambda}\delta}, \tag{4.4}$$

where z_r is the reference plane, explained in Khoshelham and Elberink, 2012, and λ is a depth camera parameter. Equation (4.4) can be written as

$$z = \frac{\lambda}{\delta + \kappa}, \tag{4.5}$$

where $\kappa = \lambda / z_r$.

Let κ' and λ' be the corrected camera parameters which are computed from κ and λ by:

$$\kappa' = \kappa + \lambda g_z, \tag{4.6}$$

$$\lambda' = \lambda(1 + h_z). \tag{4.7}$$

Let the pose of the camera with respect to the robot end-effector, determined by an inaccurate calibration procedure, be defined by a rotation matrix R and a translation vector t. The set of correction parameters is extended with a rotation vector ϕ and a vector s, which define the correction of the camera orientation and translation respectively. The corrected camera orientation is calculated as follows

$$R' = R \Delta R(\phi), \tag{4.8}$$

where $\Delta R(\phi)$ is the rotation matrix calculated from the rotation vector. In addition, the corrected camera pose is calculated as follows:

$$t' = t + s.$$

All the correction parameters discussed above can be represented by a correction vector

$$x = [g_x, g_y, g_z, h_x, h_y, h_z, \phi_x, \phi_y, \phi_z, s_x, s_y, s_z]^\top.$$

The camera parameters obtained by an imperfect calibration are referred to in this chapter as *original camera parameters*, while the parameters obtained by correcting the original parameters with the correction vector x are referred to as *corrected camera parameters*.

4.3.2 Correction of the Environment Model

The parameters of the environment model A, i.e. the vertex coordinates and the plane parameters, are computed using the camera parameters. So if the camera parameters are corrected, the parameters of the environment model are also corrected accordingly.

Correction of Vertices

Let p be a point obtained from an input RGB-D image using the original camera parameters. Assuming that these parameters are estimated with an error, this error leads to an error in the coordinates of p. The corrected point position p' can be computed from the position p by solving the following equation:

$$p' = \left(b(x) - a(x)p'\right) D(x)\,(p - t) + t + s, \tag{4.9}$$

where

$$\begin{aligned} a(x) &= g_z \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \Delta R^\top(\phi) R^\top, \\ b(x) &= a(x)(t+s) + 1 + h_z, \\ D(x) &= R \Delta R(\phi)(K+L)^{-1} K R^\top. \end{aligned}$$

Note that the parameters κ and λ do not appear in Equation (4.9), which means that the correction of a point position does not depend on the absolute values of these parameters, but only on the corrections g_z and h_z. The derivation of Equation (4.9) is provided in Appendix A.

Correction of Plane Parameters

Let n and d be the parameters of a plane computed with the original camera parameters. The equation defining the plane corresponding to the corrected camera parameters computed from n and d is

$$\eta^\top p = \rho, \tag{4.10}$$

where

$$\begin{aligned} \eta &= \left(D^{-1}(x) - t a(x)\right)^\top n + d a^\top(x), \\ \rho &= b(x) d + n^\top \left(D^{-1}(x)(t+s) - b(x) t\right). \end{aligned}$$

The unit normal vector n' and the corresponding plane offset d' can be computed from the vector η and the value ρ by normalizing Equation (4.10), i.e. by dividing η and ρ by the absolute value of η. The derivation of Equation (4.10) is provided in Appendix B.

4.3.3 Contact Constraints

A robotic tool can have a more or less complex shape. In this chapter, it is assumed that the part of the robot tool that is in contact with the surface of the target object can be modeled by a single convex polyhedron. Vertices, edges, and faces of polyhedra are called *features* in this chapter. The features associated with the robot's tool are denoted by f^t, while those corresponding to the environment are denoted by f^e.

Consider the case where a robot collides with the environment during the execution of a planned path because the environment model used for path planning is inaccurate due to inaccurate camera parameters used for its calculation. Two types of contacts between the robot tool and the environment in case of a collision are considered: (i) vertex-face contact and (ii) edge-edge contact. In the first case, a tool vertex f^t is in contact with a face f^e of an object in the environment, as shown in Figure 4.3.a). Similarly, an environment vertex can be in contact with a tool face.

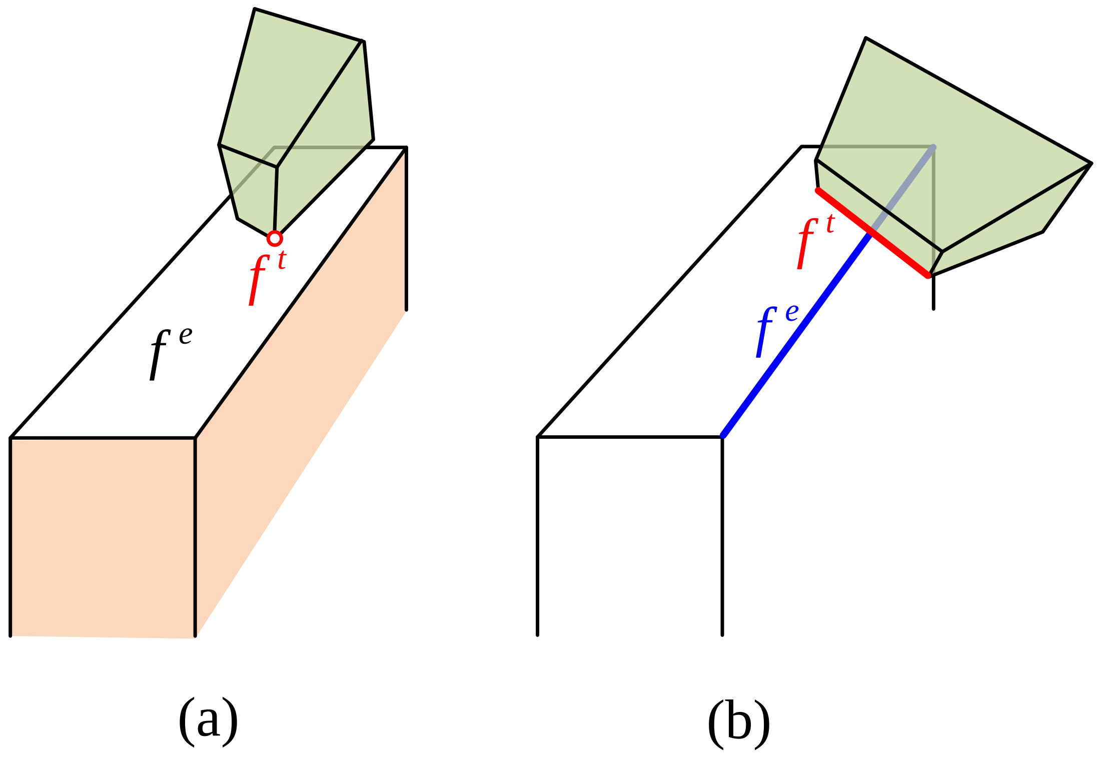

Figure 4.3: Vertex-face (a) and edge-edge (b) contact. The robot tool is painted in green and the environment object in white and orange. The orange color indicates the faces adjacent to the contact face f^e. In the case of vertex-face contact (a), the vertex of the tool, shown with a red circle, touches the white face of the environment object. In the case of edge-edge contact, the tool edge represented by the red line touches the edge of the environment object represented by the blue line.

In the second case, the tool edge f^t is in contact with an edge f^e of an object in the environment, as shown in Figure 4.3.b).

If the collision between the robot tool and the environment occurs at a position where the current environment model predicts an empty space, the camera parameters should be corrected so that the corrected environment model predicts contact with an object surface when the robot tool is at that position. The path planning algorithm can use this corrected model to plan a different path. In general, several unsuccessful actions and corrections may be required before a task is successfully completed. The contact in the k-th action is represented by a pair $c_k = (f_k^t, f_k^e)$.

Vertex-Face Contact

Consider a case in which the collision occurred during the execution of a robot task at a position for which the environment model created with the original camera parameters predicts an empty space, i.e. according to this model there is a distance between the features f^t and f^e. Let $e(c_k)$ and $\gamma(c_k)$ denote the *orthogonal distance* and the *lateral distance*, respectively, between two features. If f_k^t is a tool vertex and f_k^e is an environment face, then the orthogonal distance is defined as the distance between f_k^t and the supporting plane of f_k^e, which can be written as follows:

$$e(c_k) = n_k^\top p_k - d_k,$$

where p_k is the position of the vertex f_k^t in the step k and n_k and d_k are the parameters of the supporting plane of f_k^e. The lateral distance is defined as

$$\gamma(c_k) = \max \left\{ 0, \max_{(n,d) \in N(f_k^e)} (n^\top p_k - d) \right\},$$

where $N(f_k^e)$ is the set of planes of the adjacent faces of f_k^e. If the orthogonal projection of f_k^t onto the supporting plane of f_k^e falls within the boundaries of f_k^e, then $\gamma(c_k) = 0$.

Edge-Edge Contact

Let f_k^t and f_k^e be edges and let n_k be a unit vector perpendicular to both. The orthogonal distance between these two edges is defined as the distance between two parallel planes whose normals are parallel to n_k and each plane contains one of the edges. This distance can be written as:

$$e(c_k) = n_k^\top (p_k^t - p_k^e),$$

where p_k^t and p_k^e are the closest points of these two edges. The *lateral distance* is defined as the distance between the orthogonal projections of p_k^t and p_k^e onto a plane with normal n_k, which can be written as follows:

$$\gamma(c_k) = \left\| (I - n_k n_k^\top)(p_k^t - p_k^e) \right\|.$$

However, minimizing these distances alone is insufficient, as it does not account for the orientation of the local geometry around the edges. To address this, an intersection constraint derived from the Separating Axis Theorem for convex objects (Eberly, 2001) is employed. The constraint utilizes the separating plane Π_k, defined by the normal n_k at the contact point. This plane divides the local space into two half-spaces, as illustrated in Figure 4.4. For a contact to be physically valid, the respective objects must occupy opposite half-spaces. This condition is verified by analyzing the planar faces adjacent to each edge. For each face with normal n_f and an edge with direction vector v, a tangent vector u is computed such that it lies on the face and points outward from the edge:

$$u = n_f \times v. \tag{4.11}$$

As shown in Figure 4.4, this produces two tangent vectors for the tool (u_1^t, u_2^t) and two for the environment (u_1^e, u_2^e). These vectors are then projected onto the separating normal n_k:

$$\delta_{ee} = n_k^\top u. \tag{4.12}$$

A contact is considered valid only if the projections δ_{ee} for the tool faces and the environment faces have opposite signs. In contrast, matching signs indicate that the tool geometry penetrates the environment, resulting in the rejection of the contact candidate.

To account for noise, an angular threshold τ_{ee} is incorporated, allowing the contact to be considered valid even if one of the faces slightly crosses the boundary plane Π_k.

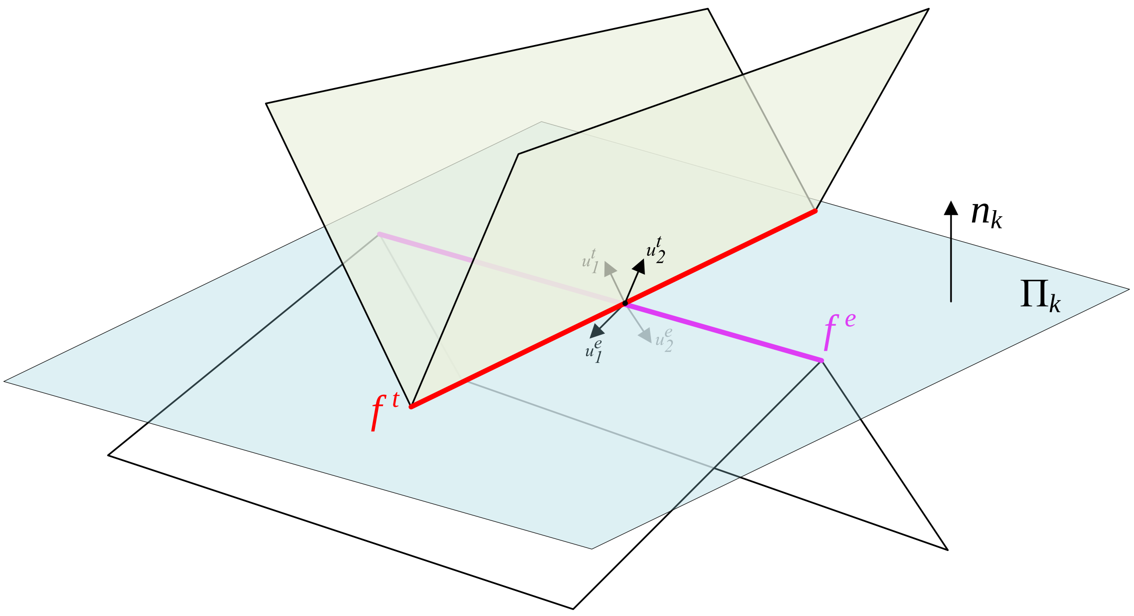

Figure 4.4: Illustration of the edge-edge intersection constraint. The cross product of the tool edge (red) and the environment edge (magenta) defines a separating plane Π_k (blue) with normal n_k. The contact is valid only if the local geometries of the tool (green) and the environment (white) occupy opposite sides of this separating plane.

Miss

Another type of failure due to an incorrect environment model occurs when the robot misses the target object, i.e. the robot misses the door panel that it should push to open the door. An illustration of such a case can be seen in Figure 4.5. According to the model of the target object represented by the dashed blue lines, the tool moving in the direction indicated by the arrows should push the target in that direction. However, since this model was created with incorrect camera parameters, the prediction of the position of the target object differs from the actual position. As a result, the tool executes the planned path without pushing the target object.

This case can be modeled by creating a virtual body that represents the smallest volume that completely contains the tool in all positions during the execution of the considered action. This body is referred to as *motion envelope*. In the example shown in Figure 4.5, the motion envelope is represented by red lines. In this chapter, it is assumed that both the tool and the target object are convex polyhedra, and only the linear motion of the tool is considered. Under these assumptions, the motion envelope is a convex polyhedron. Consider the case where the tool misses the target object during the execution of the k-th action. Since the tool has no contact with the target object during the movement under consideration, the intersection between the motion envelope and the target object must be an empty set. This is the case if there is a face f_k^t of the motion envelope for which the following equation is satisfied:

$$\min_{p \in P_{tgt}} \left(n_k^\top p - d_k \right) > 0 \,,$$

where n_k and d_k are the unit normal and the offset of the supporting plane of f_k^t and P_{tgt} is the set of all vertices of the target object. This condition is referred to as *miss constraint*.

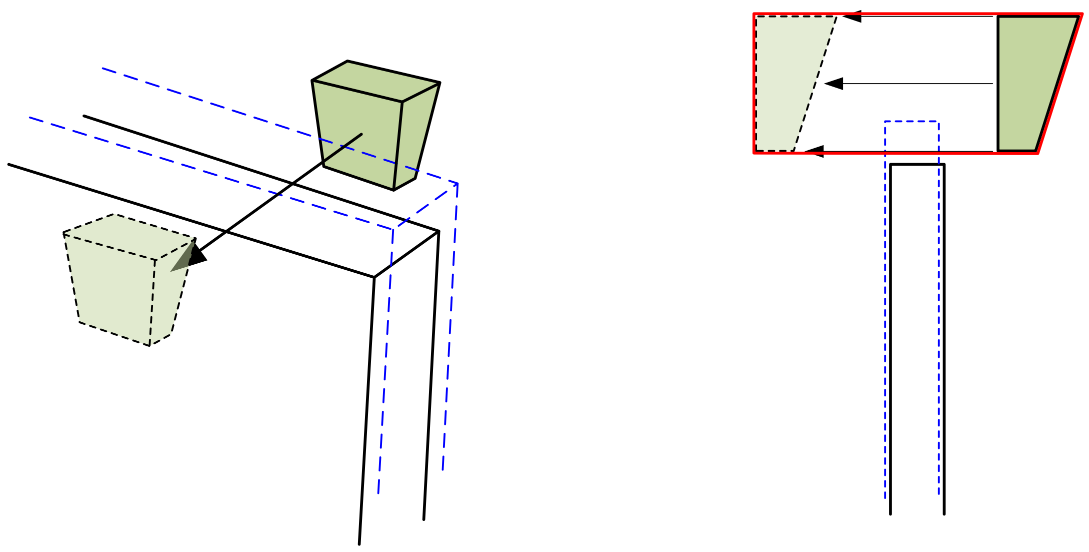
Figure 4.5: The tool (green) misses the target object. The real target object is shown with black lines, while the target object model used for action planning is shown with dashed blue lines. The arrows show the planned movement of the tool.

Let f_k^e be the vertex of the target object such that

$$p_k^e = \underset{p \in P_{tgt}}{\arg\min} \left(n_k^\top p - d_k \right)$$

where p_k^e is the position vector of f_k^e. The miss constraint can be formulated with a value called *distance-to-miss* which is defined by

$$\gamma(c_k) = \max(0, d_k - n_k^\top p_k^e).$$

If this value is greater than zero, the tool collides with the target object during its motion. The distance-to-miss can also be defined using the edge constraints for checking the intersection of convex objects introduced earlier. Analogous to the contacts, a miss is represented by a pair $c_k = (f_k^t, f_k^e)$ of features that define a miss constraint. For simplicity, the term contact is used for both contacts and misses.

For a sequence c of M contacts c_k, the camera parameters should be corrected so that, ideally,

$$e(c_k, x) = 0,$$
$$\gamma(c_k, x) = 0$$

for all k. In reality, due to the measurement noise, the optimal correction x can be computed as the vector that minimizes the values e and γ, as explained in the following section.

4.3.4 Optimal Correction

In general, the proposed method can be used to correct the camera parameters based on contacts that occur during the execution of a series of actions. It is assumed that the proposed system records an execution trace with the results of all unsuccessful actions. In case of a collision, the pose in which the collision occurred is recorded, and in case of a miss, the path segment where contact with the target should have occurred is recorded.

Actions can be performed with different target objects, captured by the robot's vision system from different viewpoints. The environment model A can include the parameters of vertices and planes of several objects, expressed in the reference frame of the robot end-effector $S_{E,\text{capture}}$, at the pose from which the target object is captured. Let c be a vector of contacts whose k-th element is the contact c_k. In this chapter, the optimal correction is computed by searching for x that minimizes the following cost:

$$E(x; A, c) = \sum_{k=1}^{M} e^2(c_k, x) + \beta \sum_{k=1}^{M} \gamma^2(c_k, x) + \alpha x^\top \Sigma^{-1} x\,, \tag{4.13}$$

for a given environment model A and contacts c, where the last term represents the squared Mahalanobis distance between the original and the corrected camera parameters. By setting the weighting factor β to a high value, it is ensured that the orthogonal projections of f_k^t and f_k^e onto a plane defined by n_k overlap. The value of α reflects the confidence in the original camera parameters. Higher values of α favor smaller corrections x. In Equation (4.13), the distances e and γ are represented as functions of a correction x, which means that they are computed using the parameters of the features f_k^t and f_k^e corrected by x. Minimization of the cost function (4.13) is performed using the Levenberg-Marquardt algorithm.

It is assumed that the robot is equipped with a sensor that can detect a collision when it occurs, but it is not known which point of the robot tool is in contact with which point of the environment. Therefore, for each robot action that is aborted due to a collision and for each miss, all possible contacts must be considered. Let θ_k be the set of all possible contacts in the case of the k-th unsuccessful action and let Θ_M be the set of all sets θ_k, $k = 1, 2, \ldots, M$. The optimal correction x can be determined by computing the optimal x for each c, where c represents a combination of possible contacts, i.e. elements of c are contacts from the set Θ_M, where the k-th element of c is an element of θ_k. The vector x^* with the minimum cost from Equation (4.13) of all vectors x computed for each c represents the optimal correction. However, the computational complexity of this algorithm would increase exponentially with the number M of actions. To overcome this problem, an optimization strategy based on random sampling of contact combinations c is proposed. Since a correction of the camera parameters is required after each failed action, the proposed algorithm is designed to be applied recursively, i.e. it uses the information of all M actions performed up to a certain point in time to compute the correction of the camera parameters. The environment model created with these corrected camera parameters is then used to plan the next action.

The proposed algorithm is shown in Algorithm 6. An optimization step performs a random sampling from the set of all possible contacts, resulting in N_{samp} contact sequences c, computes the optimal x for all of them, and returns N_{best} contact sequences c with the lowest cost along with the optimal correction x^*. The inputs for an optimization step are the environment model A, the set Θ_{M-1} of all possible contacts up to action M and the set X_{M-1} of pairs (c, x) created in the previous step. Each pair $(c, x) \in X_{M-1}$ consists of a combination of possible contacts c from actions 1 to $M - 1$, along with a correction vector x that minimizes the associated contact cost. The optimization step provides a set X_M of N_{best} pairs (c, x), where the sequences c also contain possible contacts from the action M stored in the set θ_M.

The number of contact combinations considered in the proposed optimization procedure is limited to a user-defined value N_{samp}. To steer the optimization process towards a low-cost solution, the algorithm stores a set X of N_{best} contact combinations with the lowest cost and appends to each of these combinations a contact from θ_M with

Algorithm 6 OptimizationStep

```
Input: $A, \Theta_{M-1}, \theta_M, X_{M-1}, N_{\text{samp}}, N_{\text{best}}$
Output: $x^*, \Theta_M, X_M$
 1: $C \leftarrow ContactCombinations(A, \Theta_{M-1}, \theta_M, X_{M-1}, N_{\text{samp}})$
 2: $E_{\min} \leftarrow \infty$
 3: for every $c \in C$ do
 4:     Find $x_c^*$ which minimizes $E(x; A, c)$.
 5:     if $E(x_c^*; A, c) < E_{\min}$ then
 6:         $E_{\min} \leftarrow E(x_c^*; A, c)$
 7:         $x^* \leftarrow x_c^*$
 8:     end if
 9: end for
10: $X_M \leftarrow N_{\text{best}}$ contact sequences from $C$ with lowest costs $E$ with associated corrections $x$
11: $\Theta_M \leftarrow \Theta_{M-1} \cup \theta_M$
12: return $x^*, \Theta_M, X_M$
```

the lowest error. In this way, a new set of contact combinations is obtained. The remaining $N_{\text{samp}} - |X|$ contact combinations are obtained by random selection from the set Θ_M. The proposed strategy for selecting contact combinations is presented as Algorithm 7.

Function *CorrectEnvModel* in Algorithm 7 corrects the parameters of the environment model A using the correction x.

4.4 Implementation

In this section, details on the implementation of the proposed framework for adaptive door opening and the proposed correction method are provided.

4.4.1 Door Model

The door model utilized in this chapter builds upon the models presented in Chapters 2 and 3. It has all properties described in those chapters, including reference frames S_D, $S_{A'}$, and the transformation ${}^{A'}T_D$ with an additional component introduced to represent the set of contact poses sampled on the door. This extension is discussed in detail in the following section. The complete door model is illustrated in Figure 4.6.

As described in Section 4.2.1, the model is estimated using the DDD-THD method presented in Chapter 2. The pose of the door axis S_A is expressed in the camera frame S_C as ${}^{C}T_A$. Given the eye-in-hand configuration of the camera, the relation between the camera reference frame S_C and the robot end-effector frame S_E can be defined as ${}^{E}T_C$, obtained through a calibration procedure.

The transformation between the robot end-effector in the pose corresponding to the door capture and the robot base is stored as ${}^{R}T_{E,\text{capture}}$. This pose is referred to as the *capture pose*.

4.4.2 Path Planning

It is assumed that the full kinematic model of the robot is known, allowing computation of the robot's joint configuration for a given tool pose using inverse kinematics and

Algorithm 7 ContactCombinations

```
Input: $A, \Theta, \theta, X, N_{\text{samp}}$
Output: $C'$
 1: $\Theta' \leftarrow \Theta \cup \theta$
 2: $N_{comb} \leftarrow$ number of all possible sequences $c$ created from the elements of $\Theta'$
 3: if $N_{comb} > N_{\text{samp}} - |X|$ then
 4:     $C' \leftarrow \emptyset$
 5:     for $i \leftarrow 1$ to $N_{\text{samp}} - |X|$ do
 6:         for $k \leftarrow 1$ to $M$ do
 7:             $\theta_k \leftarrow k$-th element of $\Theta'$
 8:             $c_k \leftarrow$ randomly selected element of $\theta_k$
 9:         end for
10:         $c' \leftarrow (c_1, c_2, \ldots, c_k, \ldots, c_M)$
11:         Add $c'$ to $C'$.
12:     end for
13: else
14:     $C' \leftarrow$ all possible sequences $c$ created from the elements of $\Theta'$
15: end if
16: for every $(c, x) \in X$ do
17:     $A' \leftarrow CorrectEnvModel(A, x)$
18:     for all $z = (f^t, f^e) \in \theta$ do
19:         Compute $e(z, x)$ and $\gamma(z, x)$ using the parameters of $A'$.
20:     end for
21:     $z^* \leftarrow \arg\min_{z \in \theta} (\max\{e(z, x), \gamma(z, x)\})$
22:     $c' \leftarrow (c, z^*)$
23:     Add $c'$ to $C'$
24: end for
```

conversely, determining the tool pose from a given set of joint angles using forward kinematics. Additionally, it is assumed that the pose of the fingertip of the gripper S_{TCP} in relation to the robot's end-effector S_E is known.

For a given door axis pose ${}^{C}T_{A}$ estimated by the door detection method, the door pose in relation to the robot's base S_R can be computed:

$$ {}^{R}T_{A} = {}^{R}T_{E,\text{capture}}\, {}^{E}T_{C}\, {}^{C}T_{A}. \tag{4.14} $$

This enables determining the expected motion of the door for different opening angles:

$$ {}^{R}T_{D}(\varphi) = {}^{R}T_{A}\, {}^{A}T_{A'}(\varphi)\, {}^{A'}T_{D}. \tag{4.15} $$

To plan a path for the robot's gripper to interact with the door, a set of potential contact poses ${}^{D}T_{TCP}$ is defined. These poses describe how the fingertip of the gripper S_{TCP} should be positioned and oriented relative to the point S_D. The contact poses are generated by sampling positions on a vertical line that lies on the back face of the door, shown as a dark green line in Figure 4.6. Each sampled position is combined with a range of gripper orientations, resulting in a dense set of candidate contact poses on the back face of the door. Each pose is then checked for potential collisions between the gripper and the door panel using the *Flexible Collision Library* (FCL) (Pan, Chitta and Manocha, 2012), and only the collision-free poses are retained.

Each valid contact pose is then used to construct a complete motion path consisting

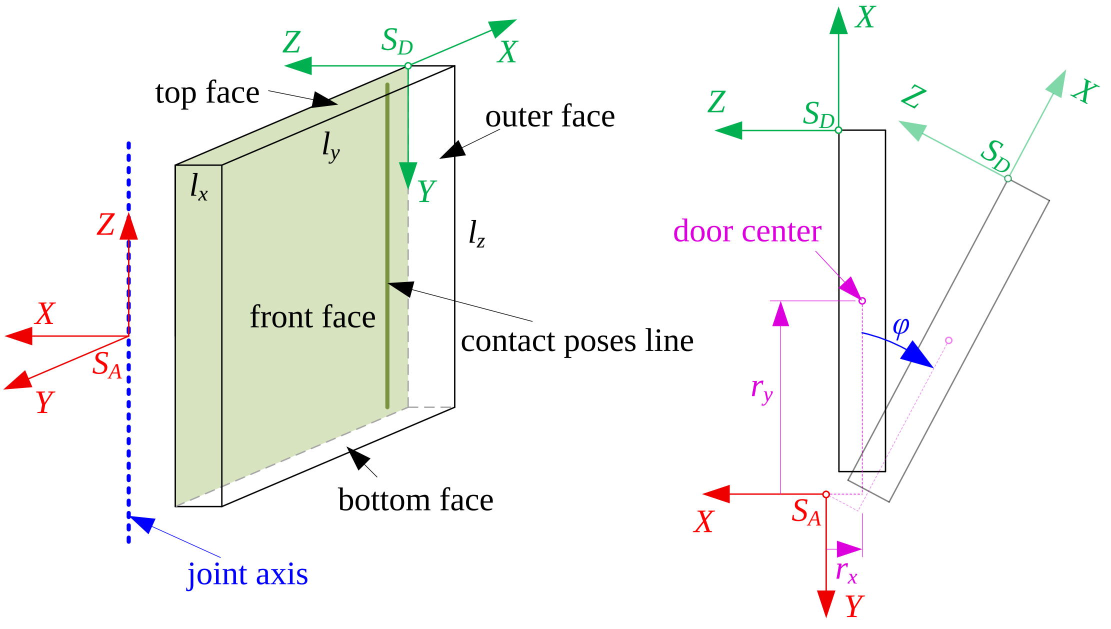

Figure 4.6: Reference frames of the door model. S_A denotes the door axis frame, while S_D refers to the reference frame of the point placed on the back face of the door panel. The back face of the door is highlighted in light green and the dark green line represents the segment along which contact positions are sampled.

of three phases: approach, insertion, and opening. Figure 4.7 illustrates the execution of the first two phases, represented by the following points: the approach point (Figure 4.7.a)), the insertion point (Figure 4.7.b)), and the contact pose (Figure 4.7.c)), where the tool establishes contact with the door. Although the robot performs the motion in this order, the approach and insertion poses are computed starting from the contact pose. To determine the insertion point, S_{TCP} is retracted away from the door contact pose in negative z-direction of S_{TCP} until a bounding sphere around the gripper no longer intersects with the planes of the cabinet's outer and top faces. From there, the gripper is further moved away from the cabinet along the negative z-direction of S_D, until the bounding sphere of the gripper no longer intersects with the plane of the front face of the door panel. This pose defines the approach point. This ensures that the robot can reach the contact point safely while maintaining the desired orientation.

The motion of the robot's end-effector during the door-opening path is represented as a sequence of poses, each corresponding to a discrete value of the door state φ, ranging from 0 to the desired goal angle. For each door opening angle φ_k, the robot's end-effector pose is computed as:

$$^{R}T_E\left(\varphi_k\right) = {}^{R}T_D(\varphi_k)\,{}^{D}T_{TCP}\,{}^{E}T_{TCP}^{-1}. \tag{4.16}$$

Joint configurations q_k corresponding to these poses are then computed using inverse kinematics. The resulting trajectory can be represented by a sequence

$$Q = (q_{\text{approach}}, q_{\text{insertion}}, q_0, \ldots, q_{n-1}), \tag{4.17}$$

where q_{approach} denotes a joint configuration for the approach point, $q_{\text{insertion}}$ denotes the configuration for the insertion point, q_k corresponds to the end-effector pose of the robot $^{R}T_E\left(\varphi_k\right)$ at a certain door state φ_k. The value n stands for the number of discrete configurations along the door opening path.

Each path is generated ensuring no collisions with the environment or self-collisions

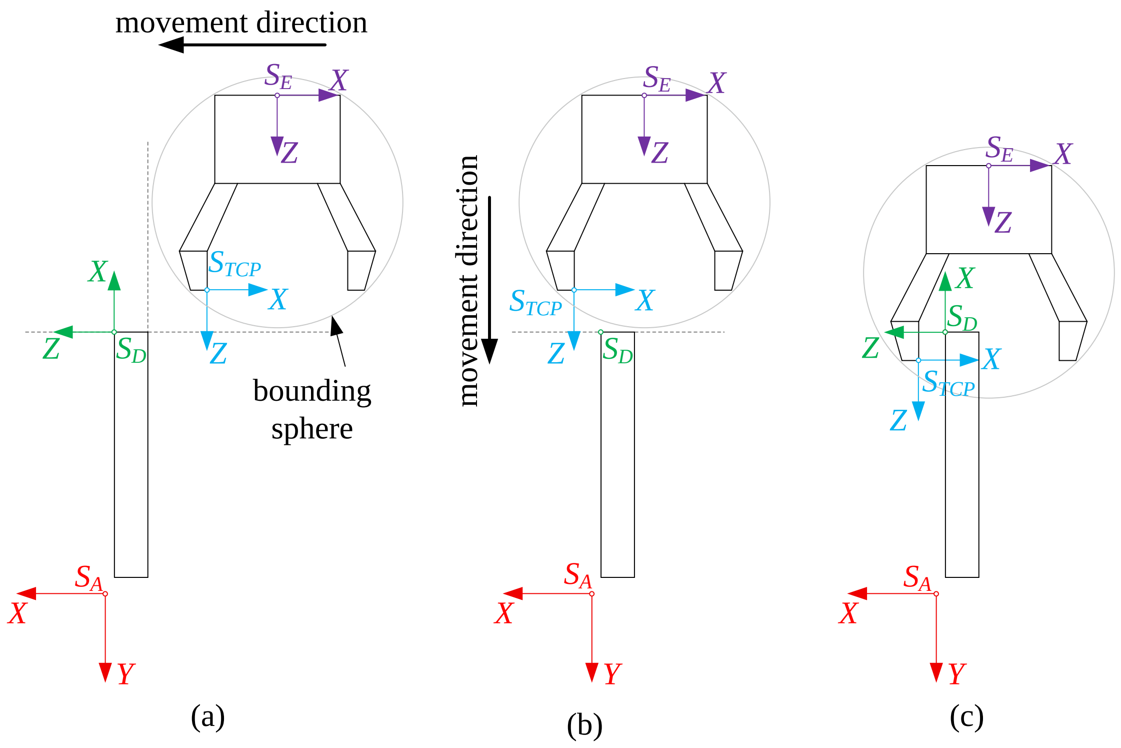

Figure 4.7: Illustration of the approach point (a), insertion point (b), and contact point (c). The insertion and approach points are generated by moving the gripper away from the door until its bounding sphere (shown in gray) no longer intersects the planes of the door's surfaces, represented by dashed gray lines.

occur and the motion remains smooth in joint space, meaning that the Chebyshev distance between the points in Q is within a threshold of 45°. In the end, the planner produces a set of feasible joint-space trajectories, each corresponding to a unique gripper contact pose.

4.4.3 Correction

In this chapter, a cabinet and its features f^e at a specific door state φ are represented as a *scene* expressed in relation to the robot end-effector in capture pose $S_{E,\text{capture}}$. A scene is created either when a new cabinet is captured and detected or when the door state φ of the existing cabinet changes. Each unsuccessful robot action is recorded as a part of an execution trace, with each action associated with the corresponding scene in which it occurred. The cabinet model and the tool model are shown in Figure 4.8.

Information about all previous scenes and actions is retained and used to correct the camera parameters in the current scene. After the correction is performed, the features of the cabinet model and the pose of the cabinet ${}^{R}T_{A}$ are updated, after which the planner can replan the path for the door opening.

4.5 Experiments

This section investigates the benefits of using the proposed approach to correct the camera parameters and the model of the target object with which the robot interacts. To this end, a series of qualitative and quantitative experiments was conducted, including simulation experiments that simulate camera errors and real-world experiments in which the

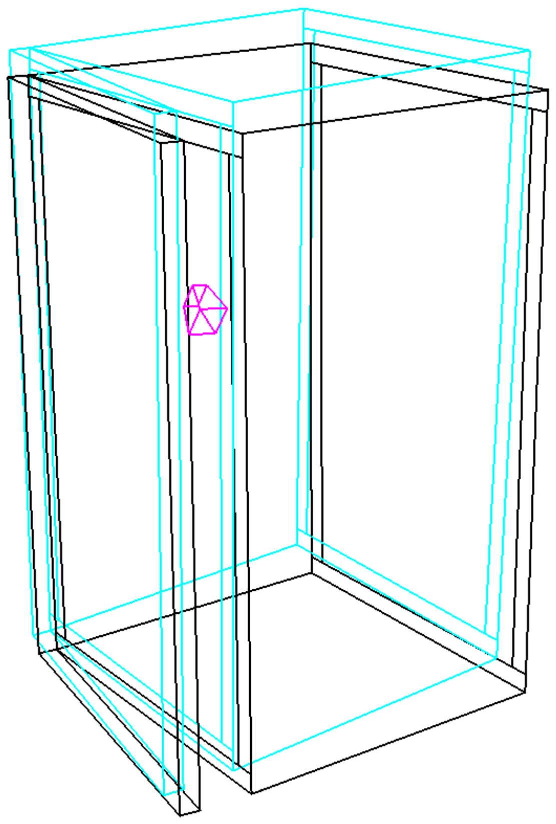

Figure 4.8: A scene representing an estimated cabinet model colored in cyan, a ground truth cabinet model colored in black and a tool model colored in magenta.

robot performed a door-opening task in its vicinity. These experiments aim to demonstrate the system's capability to adapt to perception inaccuracies and improve the overall task success rate.

4.5.1 Simulation Experiments

The purpose of the simulations was to evaluate whether the proposed method can recover from failures caused by imperfect camera calibration and progressively compensate for calibration errors during robot operation. In addition, a sensitivity analysis was performed to investigate the influence of key user-defined parameters on the performance of the approach.

The target object was a cabinet with a door, whose 3D model is shown in Figure 4.8. The object's size and pose with respect to the robot were randomly sampled from a uniform distribution that ensured the object remained within the robot's workspace. The model itself consists of polyhedral elements defined by vertex coordinates and face parameters.

The calibration error was simulated by randomly perturbing the parameters of a reference camera model and using these parameters to compute the parameters of the target object. As a reference model, the parameters of the RGB-D camera employed in the real-world experiments were used. The intrinsic camera parameters were perturbed by drawing the values g_x, g_y, g_z, h_x, h_y and h_z from normal distributions with zero mean and variances, which are shown in Table 4.1. These values were used to compute the perturbed camera parameters according to Equations (4.2), (4.3), (4.6) and (4.7). The parameters s_x, s_y and s_z were computed in the same manner using variance σ_s^2 and added to the translation vector t, which represents the position of the camera reference frame with respect to the robot end-effector. Camera orientation was perturbed by randomly selecting a point u on a unit sphere and multiplying it by $\psi \sim \mathcal{N}(0, \sigma_\phi^2)$ to obtain the vector ϕ, which was then used to compute the rotation matrix that defines the orientation of the camera reference frame with respect to the robot end-effector according to Equation (4.8).

Table 4.1 Camera perturbation parameters and their associated noise variances

Parameters	Variance	Parameters	Variance
g_x, g_y	$\sigma_{g,xy}^2$	g_z	$\sigma_{g,z}^2$
h_x, h_y	$\sigma_{h,xy}^2$	h_z	$\sigma_{h,z}^2$
ϕ_x, ϕ_y, ϕ_z	σ_ϕ^2	s_x, s_y, s_z	σ_s^2

Section 4.3.1 describes how the parameters of the target object model, i.e. the vertex coordinates and the parameters of the supporting planes of the faces of the object, can be computed from their values obtained by an imperfectly calibrated camera and a correction vector x. In the simulations, the inverse computation was applied to obtain target model parameters corresponding to a perturbed camera model. These model parameters are referred to as *estimated model parameters*, which differ from the true model parameters due to the measurement error introduced by the perturbation of the camera parameters. For a given set of perturbed camera parameters, if the maximum distance between the vertices of the estimated model and the corresponding vertices of the true target object model was greater than a threshold d_{vtx}, then this set of camera parameters was discarded and a new perturbation of the camera parameters was performed.

The estimated model parameters were used to plan a simple robot operation consisting of two steps: (i) positioning the robot tool in close proximity to the contact point with the door panel and (ii) pushing the door panel in the opening direction. The contact point chosen for each cabinet in the experiments was randomly selected from the set of feasible trajectories computed in the same way as described in Section 4.4.2. The starting point of the tool path was 0.4 m away from the contact point along the z-axis of the end-effector reference frame, and the end point was 0.4 m away from the contact point in the direction perpendicular to the door panel. The operation was considered successful if the robot tool makes contact with the back face of the door panel during the execution of the second step. If the tool collided with the door panel during the first step or missed it during the second, the operation was considered unsuccessful, and a correction was required.

The correction of the camera parameters was performed using the approach proposed in Section 4.3. The corrected camera parameters were used to compute the vertex coordinates and plane parameters of the target object model. This model was then used for planning a new path. This process was repeated until the operation was successfully completed or the limit of eight actions was reached. If the operation was not completed successfully after eight actions, the algorithm was considered to have failed to accomplish the task.

4.5.2 Real-World Experiments

To demonstrate the effectiveness of the proposed correction method in compensating for inaccuracies in camera parameters, two types of qualitative experiments in a real-world setting were conducted. The experimental setup consisted of a Universal Robots UR5 robot mounted on a table, equipped with a Robotiq FT300-S force-torque sensor, a Robotiq 3-Finger Gripper, and a XELA uSPa 44 tactile sensor installed on the fingertip of the gripper, as depicted in Figure 4.1.

Generating Cabinet Poses

For the door-opening task, a cabinet was placed on the table in different poses. These poses were randomly generated under several constraints. First, all four corners of the cabinet's bottom plate were required to remain on the table. Additionally, the cabinet's door had to avoid collision with the robot base during the opening motion. Finally, the planner needed to generate at least one opening trajectory for a given pose. The properties of the cabinet are summarized in Table 4.2.

Table 4.2 Cabinet properties in real-world experiments

	Property	Min. Value	Max. Value
Size	Width [m]	0.395	
	Height [m]	0.495	
Pose	Position in X [m]	-0.35	0.00
	Position in Y [m]	0.40	0.80
	Rotation [°]	-60	60

Door-Opening Trials

In the first experiment, the system's ability to adapt in real time without any prior corrections was evaluated. Ten cabinet poses were randomly sampled on the table, and for each pose, a random contact point on the door was selected. The robot then attempted to open the door by 45°. For the first cabinet pose, the initial action was executed based solely on the pose estimated by the door detection method. If an action failed, the robot performed a correction using the approach proposed in Section 4.3 and retried at the same location until the door was successfully opened. For subsequent cabinet poses, the initial action incorporated corrections derived from the previous scenes.

In the second experiment, the effect of an offline calibration phase prior to the door-opening task was evaluated. This phase consisted of presenting the robot with three randomly sampled cabinet poses in which the robot was required to make successful contact at three distinct locations on the back of the door panel: near the top, middle and bottom. After each failed action, the cabinet pose was corrected and the task was retried until the tactile sensor registered a contact. The resulting camera parameters were used for the door-opening task. After the offline calibration phase, the robot proceeded to open the door in the same 10 cabinet poses used in the first experiment. As before, it selected a random contact point for each pose and attempted to open the door by 45°. If an action failed, the robot corrected the pose based on the failure and retried the motion until the door was successfully opened. The purpose of this simple calibration procedure is to reduce the number of failures during the operational use of the robot. It is practical because it does not require any special equipment, but relies solely on the objects naturally present in the robot's environment.

4.6 Results

In this section, the results of the experiments described in Section 4.5 are presented and analyzed.

4.6.1 Results of Simulation Experiments

The performance of the proposed method was evaluated in simulation through a sensitivity analysis, in which each parameter was varied independently while all others were kept at their baseline values. The tested parameters are divided into two categories: camera perturbation parameters and optimization parameters. The camera perturbation parameters are used to simulate errors in the camera model and include the variances of the intrinsic parameters described in Section 4.5.1, as well as d_{vtx}, which defines the maximum allowable distance between the estimated vertices of the cabinet model and the corresponding vertices of the ground truth model. The optimization parameters correspond to the correction process. Specifically, α and N_{best} were evaluated, both of which are described in Section 4.3.4. The full set of tested values for each parameter is summarized in Table 4.3, where the bold value in each range denotes the baseline configuration used in the experiments.

Table 4.3 User-defined parameters used in the sensitivity analysis and their tested values. The bolded value in each range was used as the baseline.

Parameter	Tested Values
Camera Perturbation Parameters	
$\sigma_{g,xy}$	$\{2\%, \mathbf{5}\%, 10\%\}$
$\sigma_{g,z}$	$\{1\%, \mathbf{2}\%, 5\%\}$
σ_{ϕ}	$\{1^\circ, \mathbf{2}^\circ, 3^\circ\}$
d_{vtx}	$\{\mathbf{4}\text{ cm}, 5\text{ cm}, 6\text{ cm}\}$
Optimization Parameters	
α	$\{10^{-4}, 10^{-5}, \mathbf{10^{-6}}, 10^{-7}\}$
N_{best}	$\{0, 50, 100, \mathbf{500}, 750\}$

The evaluation was performed on a total of 1000 cabinet configurations, split into 50 sequences of 20 scenes each. At the beginning of each sequence, the camera parameters were randomly perturbed, as explained in Section 4.5.1, and the execution trace was emptied. If an action in a given scene resulted in failure, the correction algorithm was executed while considering all failed actions from the start of the scene sequence to which that scene belonged.

In Table 4.4, the results of the simulation experiments are presented. The proposed method exhibits high robustness across a wide range of parameter settings. Despite the overall high success rates, a small number of cabinet-opening tasks failed to complete even after 8 actions. These cases are reported as *Collisions* and *Misses* in Table 4.4. Collisions refer to failures where, even after 8 unsuccessful actions, the robot made unintended contact with the cabinet while positioning the tool for opening. On the other hand, misses describe instances where the tool failed to make a contact with the door panel during the opening motion.

For the camera perturbation parameters, the method maintained a success rate above 99% in all tested configurations, even with significant perturbations such as $\sigma_{g,xy} = 10\%$ and $\sigma_{\phi} = 3^\circ$. This indicates that the correction method is capable of recovering from failures for significant inaccuracies in the estimated camera model. For the optimization parameters, most configurations also achieved a success rate above 99%. However, removing the set X of the best contact combinations with lowest cost and only relying on the random selection of possible contacts ($N_{\text{best}} = 0$), led to a significant drop in performance. This suggests that reusing previously found low-cost solutions helps guide the optimization toward improved solutions. In addition, there was a slight drop in performance at a higher value of $\alpha = 10^{-4}$, suggesting that over-reliance on the original camera parameters limits the method's ability to compensate for larger errors.

On average, there were 5.9 collision cases across all configurations. Among the camera perturbation parameters, the largest number of collisions occurred with $d_{\text{vtx}} = 5$ cm and $d_{\text{vtx}} = 6$ cm, indicating that increasing the error in reconstruction of the simulated model can lead to more unsuccessful contact attempts. For the optimization parameters, a significant rise in failure rates due to collision was observed when biasing the optimization heavily on initial camera parameters ($\alpha = 10^{-4}$) or removing contacts that previously produced low-cost solutions ($N_{\text{best}} = 0$). By contrast, misses were rare across most configurations, with the majority resulting in zero missed tasks. However, a notable increase in missed instances can be seen for $N_{\text{best}} = 0$, where 6 tasks resulted in a miss.

Table 4.4 Success metrics for each configuration across 1000 simulated cabinet-opening tasks

Configuration	**Successes**	**Collisions**	**Misses**	**Success Rate [%]**
Camera Perturbation Parameters				
Baseline	996	4	0	99.6
$\sigma_{g,xy} = 2\%$	998	2	0	99.8
$\sigma_{g,xy} = 10\%$	997	3	0	99.7
$\sigma_{g,z} = 1\%$	999	1	0	**99.9**
$\sigma_{g,z} = 5\%$	997	3	0	99.7
$\sigma_{\phi} = 1^\circ$	995	4	1	99.5
$\sigma_{\phi} = 3^\circ$	998	2	0	99.8
$d_{\text{vtx}} = 5$ cm	990	8	2	99.0
$d_{\text{vtx}} = 6$ cm	995	5	0	99.5
Optimization Parameters				
Baseline	996	4	0	99.6
$\alpha = 10^{-4}$	988	12	0	98.8
$\alpha = 10^{-5}$	997	3	0	**99.7**
$\alpha = 10^{-7}$	996	4	0	99.6
$N_{\text{best}} = 0$	964	30	6	96.4
$N_{\text{best}} = 50$	994	6	0	99.4
$N_{\text{best}} = 100$	996	4	0	99.6
$N_{\text{best}} = 750$	996	4	0	99.6

The influence of parameter selection on the effectiveness of the proposed method is also analyzed in terms of the required number of actions for a successful door opening.

Figure 4.9 shows the distribution of the number of actions required for a successful task across 1000 scenes for each parameter in simulation. The top plot presents different values of the camera perturbation parameters, while the bottom plot presents different values of the optimization parameters, where each bar indicates the number of scenes that required a specific number of actions for a given configuration. All configurations demonstrated that the task was successfully completed within 2 actions in over 90% of the cases. However, introducing higher errors into the camera model, such as $\sigma_{g,xy} = 10\%$, $\sigma_{g,z} = 5\%$ or $d_{\text{vtx}} = 6$ cm, resulted in a slightly higher number of required actions. Notably, setting $N_{\text{best}} = 0$ led to a marked decrease in successes in the first action and a corresponding rise in cases requiring 4 or more actions.

For all cabinets in the simulation, the number of required actions averaged across 50 sequences of 20 scenes is analyzed, as shown in Figure 4.10. In the first scenes, the number of actions before success was high due to the simulated error in the camera parameters, which has not yet been corrected. In most cases, this number gradually decreased as the robot collected data from the collisions and misses. Despite increasing the variance of the camera parameters, the performance of the method remained consistent across most configurations. By increasing the vertex threshold d_{vtx} to 6 cm, the number of required actions initially increased too, but over time it gradually decreased and reached the same level as other configurations. Similar effect was observed for $\alpha = 10^{-4}$. In contrast, by setting $N_{\text{best}} = 0$, while the number of actions was initially similar to other configurations, it slowly started increasing as the number of the randomly selected contacts do not necessarily provide a low-cost solution that would help decrease the required number of actions.

Figure 4.11 shows the distribution of actions before success for each of 20 tested scenes for the baseline configuration. Within each group of bars corresponding to a scene, the individual bars represent the number of actions required to complete the task. For clarity, the results are categorized into tasks completed in 1, 2, or 3 actions, while all cases requiring 4 or more actions are grouped under the label 4+. It can be observed that in Scene 1, the method exhibited the lowest number of successes on the first try accompanied by a higher frequency of multiple required actions. However, as the simulation progressed, the frequency of first-action successes increased, while the number of tasks requiring multiple actions declined, especially from Scene 5 onward. This improvement aligns with the intended effect of the correction mechanism, which accumulates information over time to reduce errors produced by the perception system.

4.6.2 Results of Real-World Experiments

To demonstrate the effectiveness of the proposed door-opening framework and correction method, real-world experiments were conducted using the baseline configuration described in Section 4.6.1.

In the experiments, the robot successfully opened the door in all 10 door-opening scenes, achieving a 100% success rate with both correction strategies. However, the number of actions required to complete the task varied across scenes. Figure 4.12 shows the number of actions required to successfully open the door in each scene, comparing the two strategies: with and without the offline calibration phase. In the experiment with offline calibration, the robot executed 26 actions during three calibration scenes, which served as the offline calibration phase. These initial actions included multiple collisions with the door and missed contacts. Following this phase, the robot exhibited a significant improvement in performance, successfully opening the door on the first action in 9 out of the 10 door-opening scenes. In contrast, the experiment without offline calibration required slightly more actions per scene and succeeded on the first try in 6 out of 10

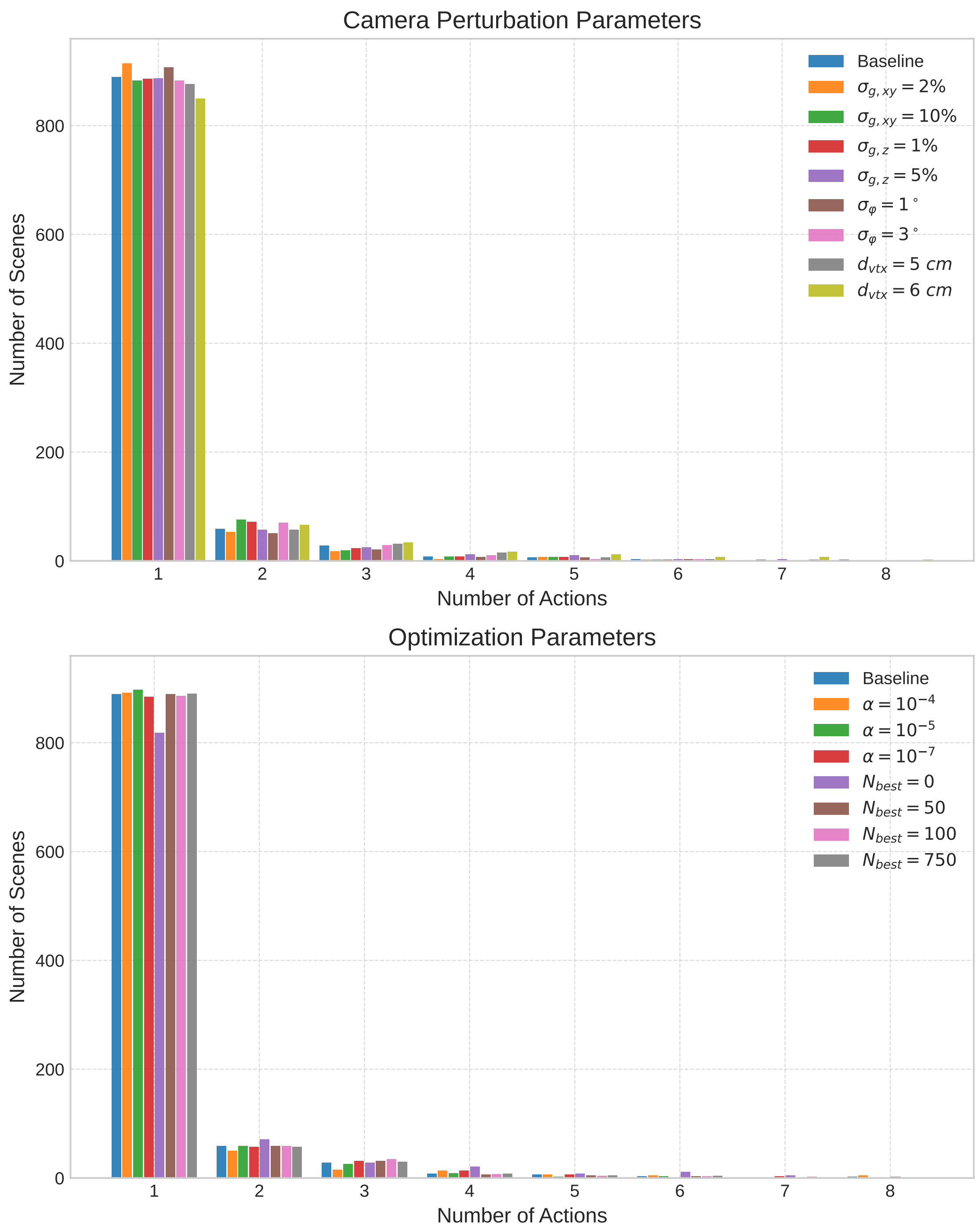

Figure 4.9: Distribution of the number of actions required for a successful task across 1000 cabinets for each tested configuration in simulation.

door-opening scenes. Regardless, the number of actions decreased over time, suggesting that the system was able to adapt to the initial errors during task execution. These results highlight the benefits of offline calibration, particularly in reducing the number of failed actions in the early stages of door-opening tasks.

To evaluate the accuracy of the corrected door pose in each scene, Euclidean distance between the origin of S_D on the corrected door panel and the corresponding point on the ground truth model was measured. This distance was recorded at the start of each new scene, before any action was made to open the door. As shown in Figure 4.13, both

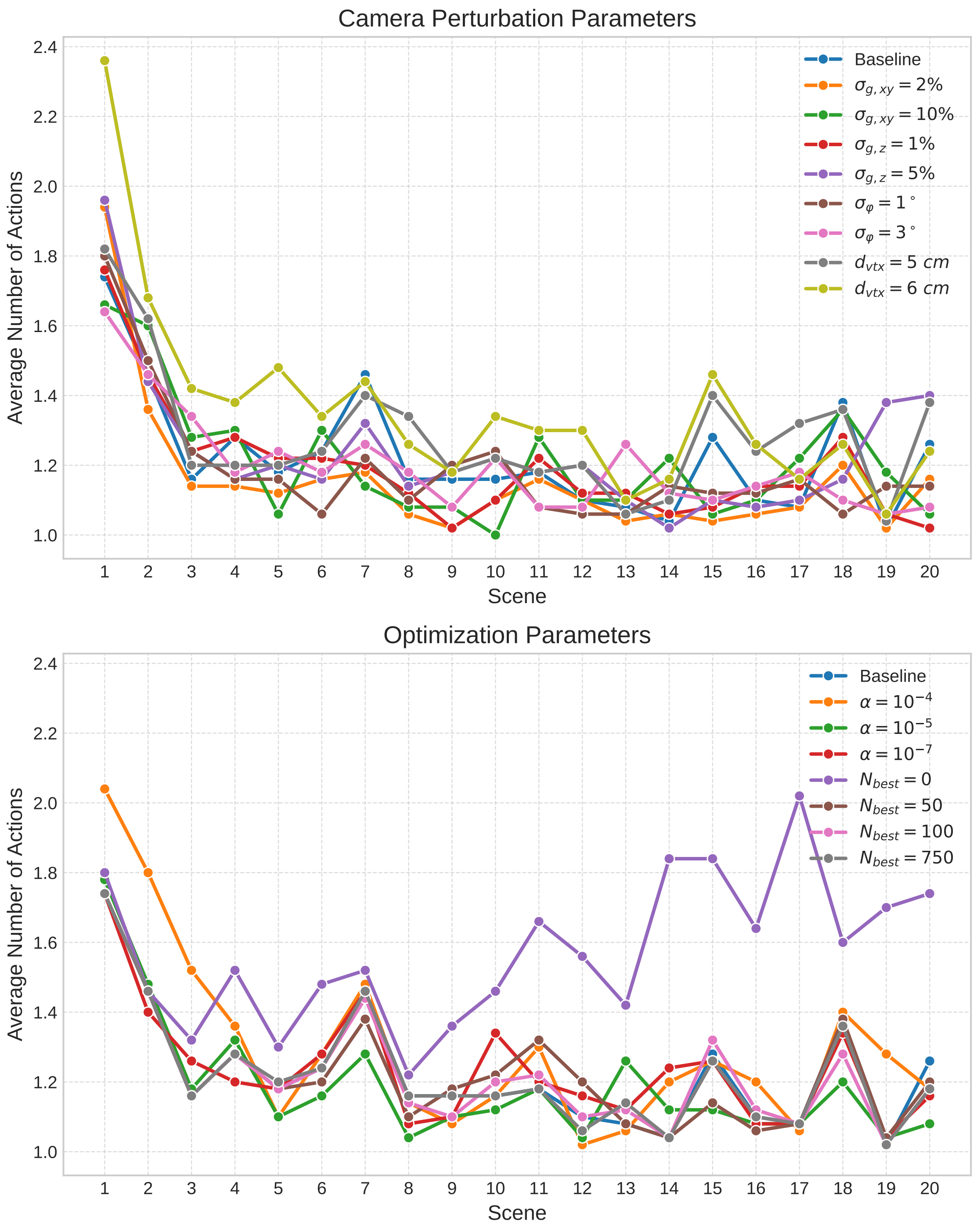

Figure 4.10: Average number of actions per scene across 20 scenes used in the simulation experiments.

strategies achieved average distances below 3 cm. In the experiment with the offline calibration, the estimated cabinet model was more consistent with the ground truth. This is due to the correction parameters remaining unchanged when the action leads to a successful task. In contrast, the experiment without the offline calibration phase produced a slightly higher average distance, with larger variation in early scenes where most of the unsuccessful actions occurred. In Scene 5, the large error in distance comes from an error in the estimation of the door height during the detection. Nevertheless, this error did not prevent successful task execution, largely due to the gap between the door

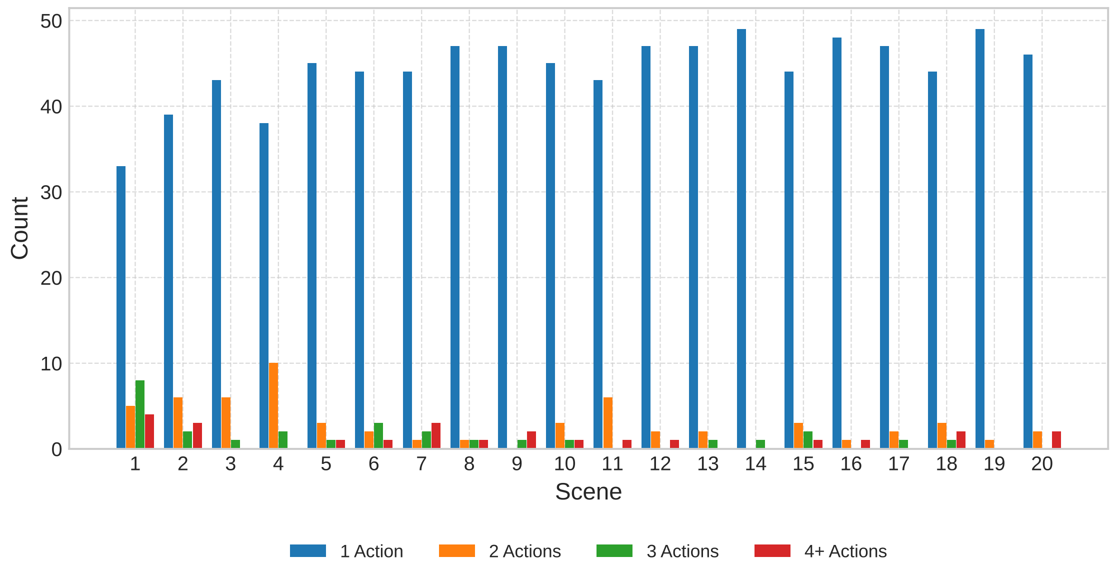

Figure 4.11: Distribution of actions per scene for the baseline configuration in the simulation experiments.

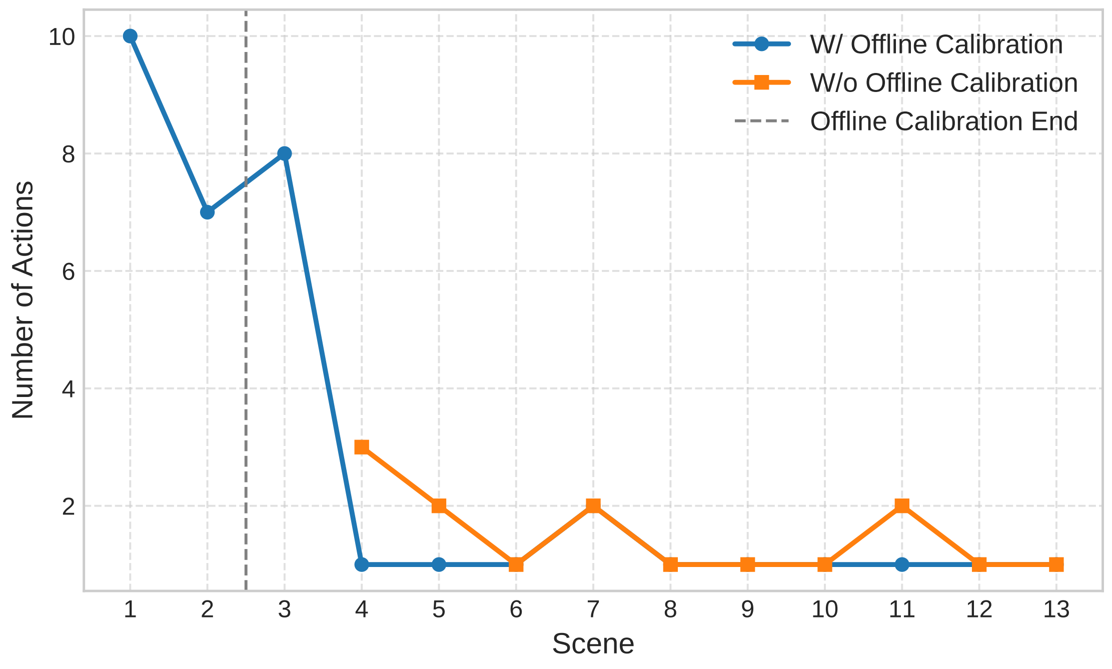

Figure 4.12: Number of required actions per scene for the real-world experiment with and without an offline calibration phase.

and the static part of the cabinet, which allows a tolerance when inserting the finger of the gripper. It is worth noting that, in some cases, the task can still be successfully accomplished despite inaccuracies in the estimated target position, provided that the error lies in a direction that does not impede task execution. If the estimation error lies along a direction that does not cause failures, the proposed method does not attempt to correct the camera parameters.

Figure 4.14 shows an example of a successful real-world door-opening experiment. The first three images depict the robot following the approach path, followed by the

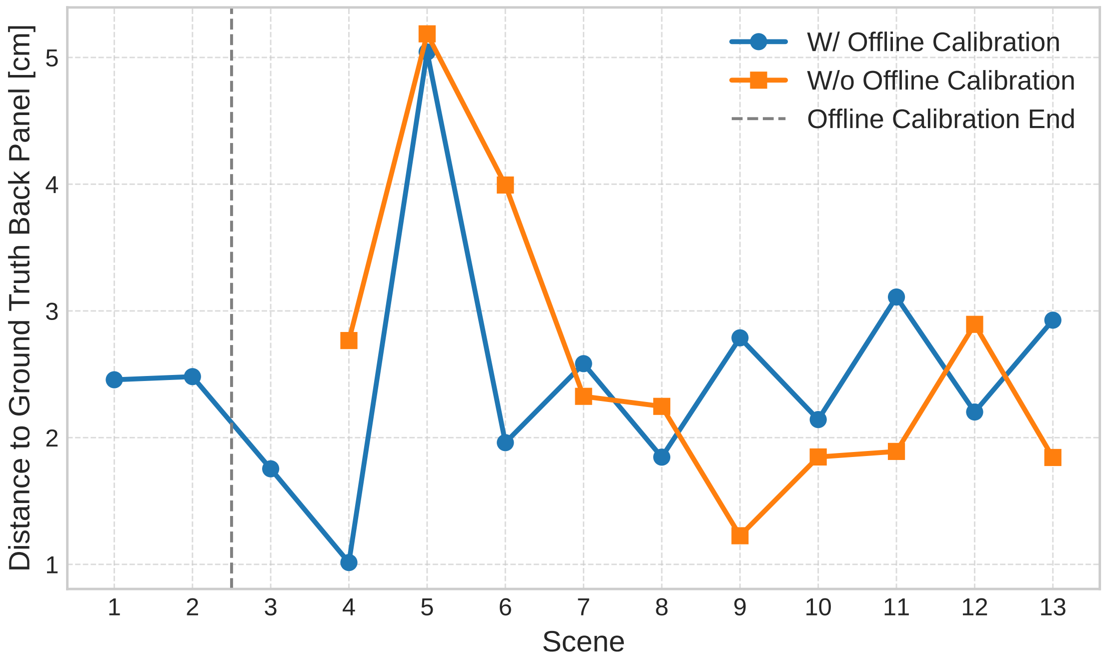

Figure 4.13: Euclidean distance between the origin of S_D on the corrected door panel and the ground truth in the real-world experiments.

insertion motion. The robot then proceeds through a planned sequence of poses to pull the door open.

4.7 Conclusion

This chapter presented a framework for opening handleless cabinets that integrates visual, force, and tactile modalities to enable efficient failure recovery. This was achieved with a novel correction method that can compensate for camera-related inaccuracies by leveraging sensory data gathered during unsuccessful robot actions. Specifically, the method uses force and tactile feedback to estimate corrections to the camera parameters, which are then used to update the cabinet model and replan a more reliable opening trajectory. The proposed approach was published in Šimundić et al., 2026.

To evaluate the framework and the proposed correction method, a sensitivity analysis was conducted using a series of simulated cabinet-opening tasks. The results showed that the correction mechanism improved performance over time, even under different levels of camera perturbation. Real-world experiments further validated the approach under realistic conditions, confirming its ability to recover from perception errors and adapt to new cabinet configurations.

One limitation of the current implementation is the lack of specificity in interpreting contacts. Since the precise location of contact between the robot tool and the environment is not known, the system must consider all possible contact candidates, which can be computationally expensive. Incorporating the force vector from the force-torque sensor at the moment of failure could help reject unlikely candidates, reducing the set of possible contacts and improving efficiency. Furthermore, utilizing tactile sensors that cover the entire gripper surface and provide precise contact point localization would significantly help the system to identify the exact contact location. While the current system

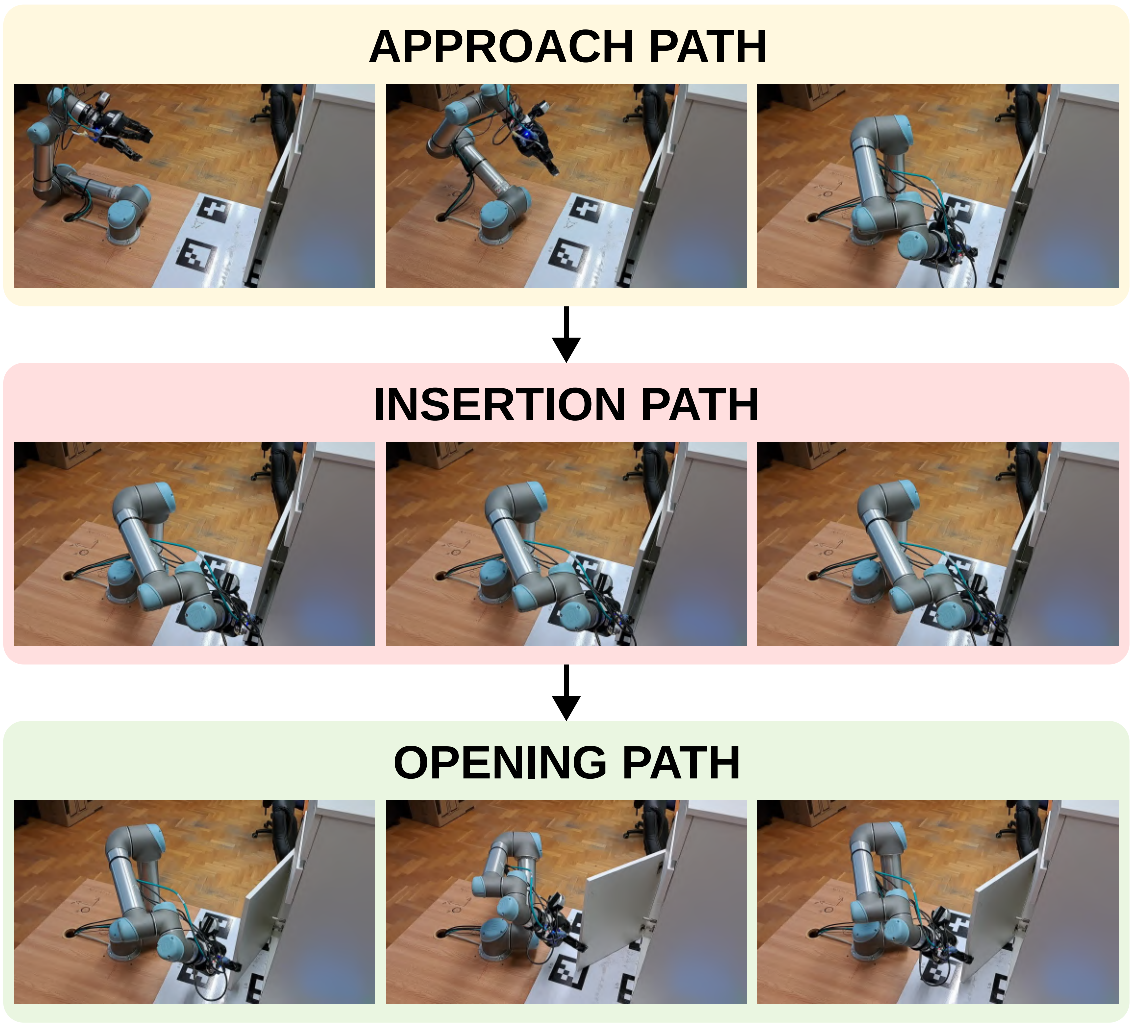

Figure 4.14: A sequence of images showing the door-opening task in a real-world experiment.

is tailored specifically for handleless doors, future work could explore adaptations of the method and the framework to other types of constrained manipulation tasks.

5 Conclusion

This thesis addressed robust interaction with articulated furniture in human-centered environments, with emphasis on opening handleless cabinet doors. The main challenge in such interactions is that successful manipulation depends on a reconstructed model of the object and its state, yet this model can be imperfect due to sensor noise, calibration inaccuracies, and perception errors. To achieve reliable interaction despite these limitations, the thesis developed a pipeline consisting of perception, multi-contact path planning, and failure recovery based on multimodal sensing.

The first part of the thesis addressed the perception of articulated objects and presented a method for door and drawer detection from RGB-D image sequences of human demonstrations and a method for state estimation using a single RGB-D image. The proposed methods provide object information used by the subsequent manipulation methods. The evaluation of both methods on real-world data demonstrated that the proposed detection and state estimation were able to reconstruct models of multiple objects with varying dimensions in different settings and configurations. The experiments also highlighted limitations that are common in RGB-D perception, including reduced detection and state estimation performance under occlusions, small observable motion, and challenging viewing configurations. These limitations can be mitigated by selecting more informative viewpoints during observation and by incorporating RGB-based or learning-based perception when depth information is unreliable.

The second part of the thesis introduced a path planning approach for opening handleless cabinet doors that considers multiple possible contact configurations, rather than relying on a single contact throughout the opening motion. The method constructs a set of feasible end-effector poses on the door surface and searches for a door-opening path that remains feasible under kinematic and collision constraints. Experimental evaluation in simulation and real-world experiments demonstrated that this approach increases the number of feasible solutions and enables reliable door opening across a wide range of cabinet poses and configurations. The evaluation further showed that the approach can improve door-opening performance even when handles are present, outperforming single-contact planning. The results also indicated that performance is sensitive to slight inaccuracies in object geometry or pose, which can lead to failed opening attempts. These observations underline the importance of monitoring interaction feedback during execution to detect deviations from the planned contact and to update the object model for subsequent attempts, which is addressed in the final part of this thesis.

The final part of the thesis addressed robust real-world door opening by integrating visual, force, and tactile sensing for failure detection and recovery. A door-opening framework was introduced in which data from force-torque and tactile sensors is monitored and failures such as collisions, missed contact, or contact loss trigger a correction procedure. The proposed method uses the tool pose at failure together with force and tactile feedback to update the camera calibration. Refining the calibration improves the reconstructed cabinet model used for planning and enables more reliable replanning in subsequent attempts. The evaluation in simulation demonstrated that the correction mechanism can progressively compensate for calibration perturbations and reduce the number of actions required over successive scenes. Real-world experiments further confirmed that the system can adapt to model and pose inaccuracies and reliably open the door across different cabinet poses. A remaining limitation is that the system does not directly observe where contact occurs during a failure. This means it cannot determine which point or region of the tool is in contact with which point or region of the environment. As a result, the recovery module must consider a large number of candidate contacts, which increases computational cost. In future work, this limitation could be mitigated by using the measured force direction to reject unlikely contact hypotheses and by employing tactile sensing with larger coverage to better identify the contact location.

Several directions can further extend the work presented in this thesis. A natural next step is a tighter integration of the multi-contact path planning method presented in Chapter 3 with the door-opening framework introduced in Chapter 4. Such an integration would result in a system that can handle a wider range of cabinet poses and react to unexpected events during execution. Furthermore, the core principles of the methods presented in this thesis are applicable beyond door opening. In particular, considering multiple contacts and adjusting camera parameters based on interaction feedback can be transferred to other contact-rich manipulation tasks. Finally, learning-based methods provide a promising complementary direction. Reinforcement learning could be used to select contact locations, choose recovery actions, or learn policies for updating camera parameters. Combining such methods with the planning and correction principles proposed in this thesis could improve robustness when transferring the approach to other contact-rich manipulation tasks.

Appendix A — Correction of Environment Vertices

The coordinates of a point p in the input RGB-D image with respect to the camera reference frame S_C are computed from the image coordinates of this point, which can be represented by a vector $m \in \mathbb{R}^2$, the depth δ assigned to this point and the camera parameters listed above according to the following equation:

$$^{C}p = {^{C}z}\,K^{-1}\,\tilde{m}, \tag{A.1}$$

where $\tilde{m}$ denotes the homogeneous image coordinates and ^{C}z the z-coordinate of the point with respect to the camera reference frame S_C. The coordinates of the same point with respect to the robot end-effector reference frame S_E can be computed using

$$p = R\,{^{C}p} + t. \tag{A.2}$$

Equation (4.9) is derived as follows. From Equation (A.1) and Equation (A.2) follows:

$$p = {^{C}z}RK^{-1}\tilde{m} + t \tag{A.3}$$

Analogously, the point coordinates computed using corrected camera parameters are:

$$p' = {^{C}z'}R'K'^{-1}\tilde{m} + t'. \tag{A.4}$$

From Equation (A.3) and Equation (A.4) the following equation can be obtained:

$$p' = \frac{^{C}z'}{^{C}z}R'K'^{-1}KR^{T}(p - t) + t'. \tag{A.5}$$

Equation (A.5) can be written as:

$$p' = \frac{^{C}z'}{^{C}z}D(x)(p - t) + t'. \tag{A.6}$$

Now, the quotient $^{C}z'/{^{C}z}$ can be expressed using correction parameters g_z and h_z. From Equation (4.5) follows:

$$\delta = \frac{\lambda}{z} - \kappa. \tag{A.7}$$

An analogous equation written for the corrected camera parameters is:

$$\delta = \frac{\lambda'}{z'} - \kappa'. \tag{A.8}$$

From Equation (A.7) and Equation (A.8) the following equation can be obtained:

$$\frac{\lambda}{{}^{C}z} - \kappa = \frac{\lambda'}{{}^{C}z'} - \kappa',$$

which can be written as:

$$\frac{{}^{C}z'}{{}^{C}z} = \frac{\lambda'}{\lambda} + \frac{\kappa - \kappa'}{\lambda}{}^{C}z'. \tag{A.9}$$

Using the definitions of the correction parameters g_z and h_z, Equation (A.9) can be written as:

$$\frac{{}^{C}z'}{{}^{C}z} = (1 + h_z) - g_z{}^{C}z'. \tag{A.10}$$

By substituting Equation (A.10) into Equation (A.6), the following is obtained:

$$p' = \left((1 + h_z) - g_z{}^{C}z'\right) D(x)(p - t) + t + s. \tag{A.11}$$

Since ${}^{C}z'$ depends on p', Equation (A.11) can be written in such a way that ${}^{C}z'$ is avoided. Now, the term that expresses ${}^{C}z'$ using p' can be derived. The value ${}^{C}z'$ is the third component of the vector ${}^{C}p'$, which represents the position of the point p with respect to the camera reference frame S_C computed using the corrected camera parameters, i.e.

$${}^{C}z' = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} {}^{C}p'. \tag{A.12}$$

The vector ${}^{C}p'$ is obtained by transforming p' from S_E to S_C. This can be expressed by the following equation:

$${}^{C}p' = \Delta R^T(\phi) R^T(p' - t - s). \tag{A.13}$$

By substituting Equation (A.13) into Equation (A.12), the following is obtained:

$${}^{C}z' = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \Delta R^T(\phi) R^T(p' - t - s). \tag{A.14}$$

By substituting Equation (A.14) into Equation (A.11), Equation (4.9) is obtained.

Appendix B — Correction of Environment Plane Parameters

The derivation of Equation (4.10) is as follows. From Equation (4.9), the following equation can be obtained:

$$p = \frac{1}{b(x) - a(x)p'} D^{-1}(x) \left(p' - (t + s)\right) + t. \tag{B.1}$$

From Equation (B.1) and Equation (4.1) follows that

$$\frac{1}{b(x) - a(x)p'} n^T D^{-1}(x) \left(p' - (t + s)\right) + n^T t = d. \tag{B.2}$$

Equation (B.2) can be written as Equation (4.10).

Curriculum Vitae

Valentin Šimundić was born in 1998 in Osijek, Croatia. He completed his primary education at Josip Antun Ćolnić Elementary School and graduated from Antun Gustav Matoš Gymnasium, both in Đakovo. In 2019, he received a Bachelor's degree in Computer Engineering from the Faculty of Electrical Engineering, Computer Science, and Information Technology Osijek (FERIT), J. J. Strossmayer University of Osijek, Croatia. In 2021, he earned a Master's degree in Computer Engineering from the same faculty, specializing in Robotics and Artificial Intelligence. He also received the Rector's Award for his professional paper titled *Sigurnosni sustav za industrijske robote zasnovan na prepoznavanju ljudi pomoću RGB-D kamere i umjetne inteligencije*, developed within the *Humans Detected by Robots* project. After enrolling in the Postgraduate Doctoral Study in Electrical Engineering and Computer Science at FERIT, he was employed as a teaching and research assistant on the Croatian Science Foundation project *Real-World Robot Perception: Understanding of Complex Scenes* (COSPER). His research focuses on robotic vision and manipulation.

Valentin Šimundić
in Osijek, Croatia
19 February 2026

Bibliography

Ahmad, Faseeh et al. (2024). 'Adaptable recovery behaviors in robotics: a behavior trees and motion generators (btmg) approach for failure management'. In: *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, pp. 1815–1822.

Ak, Abdullah Cihan, Eren Erdal Aksoy and Sanem Sariel (2023). 'Learning failure prevention skills for safe robot manipulation'. In: *IEEE Robotics and Automation Letters* 8.12, pp. 7994–8001.

Aldoma, Aitor et al. (2016). 'A Global Hypothesis Verification Framework for 3D Object Recognition in Clutter'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.7, 1383—1396. DOI: 10.1109/TPAMI.2015.2491940.

Arduengo, Miguel, Carme Torras and Luis Sentis (2021). 'Robust and Adaptive Door Operation with a Mobile Robot'. In: *Intelligent Service Robotics* 14.3, pp. 409–425.

Axelrod, Benjamin and Wesley H Huang (2015). 'Autonomous Door Opening and Traversal'. In: *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, pp. 1–6.

Banerjee, Nandan et al. (Nov. 2015). 'Human-Supervised Control of the ATLAS Humanoid Robot for Traversing Doors'. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. Seoul, South Korea: IEEE, pp. 722–729. ISBN: 978-1-4799-6885-5. DOI: 10.1109/HUMANOIDS.2015.7363442.

Berenson, Dmitry, Siddhartha Srinivasa and James Kuffner (2011). 'Task Space Regions: A Framework for Pose-Constrained Manipulation Planning'. In: *The International Journal of Robotics Research* 30.12, pp. 1435–1460.

Bohlin, Robert and Lydia E Kavraki (2000). 'Path Planning Using Lazy PRM'. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE, pp. 521–528.

Burgess-Limerick, Ben, Chris Lehnert Jurgen Leitner and Peter Corke (2023). 'Enabling failure recovery for on-the-move mobile manipulation'. In: *arXiv preprint arXiv:2305.08351*.

Byravan, Arunkumar et al. (2017). *SE3-Pose-Nets: Structured Deep Dynamics Models for Visuomotor Planning and Control*. arXiv: 1710.00489 [cs.RO]. URL: https://arxiv.org/abs/1710.00489.

Caddeo, Gabriele M et al. (2023). 'Collision-aware in-hand 6d object pose estimation using multiple vision-based tactile sensors'. In: *arXiv preprint arXiv:2301.13667*.

Calinon, Sylvain et al. (June 2010). 'Learning and Reproduction of Gestures by Imitation'. In: *IEEE Robotics & Automation Magazine* 17.2, pp. 44–54. ISSN: 1558-223X. DOI: 10.1109/MRA.2010.936947. (Visited on 03/05/2025).

Chen, Hongyi et al. (2024). 'Automating robot failure recovery using vision-language models with optimized prompts'. In: *arXiv preprint arXiv:2409.03966*.

Chen, Xinlei et al. (2019). 'TensorMask: A Foundation for Dense Object Segmentation'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Chen, Zhichao and Stanley T. Birchfield (June 2008). 'Visual Detection of Lintel-Occluded Doors from a Single Image'. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8. DOI: 10.1109/CVPRW.2008.4563142. (Visited on 02/05/2025).

Chitta, Sachin, Benjamin Cohen and Maxim Likhachev (2010). 'Planning for Autonomous Door Opening with a Mobile Manipulator'. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1799–1806.

Chung, Woojin et al. (2009). 'Door-Opening Control of a Service Robot Using the Multifingered Robot Hand'. In: *IEEE Transactions on Industrial Electronics* 56.10, pp. 3975–3984.

Cupec, R., D. Filko and E.K. Nyarko (2017). 'Segmentation of depth images into objects based on local and global convexity'. In: *Proc. European Conference on Mobile Robots (ECMR)*.

Cupec, R. et al. (2020). 'Object recognition based on convex hull alignment'. In: *Pattern Recognition* 102.

Cupec, R. et al. (Aug. 2023). 'Teaching a Robot Where Doors and Drawers Are and How To Handle Them'. In: *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 2288–2294. DOI: 10.1109/RO-MAN57019.2023.10309560. (Visited on 03/10/2025).

Cupec, Robert and Petra Đurović (2018). 'Volume Net: Flexible Model for Shape Classes'. In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 248–255.

Ding, Zihan et al. (2021). 'Sim-to-Real Transfer for Robotic Manipulation with Tactile Sensory'. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6778–6785.

Driess, Danny et al. (2023). *PaLM-E: An Embodied Multimodal Language Model*. arXiv: 2303.03378 [cs.LG]. URL: https://arxiv.org/abs/2303.03378.

Eberly, David (2001). *Testing for Intersection of Convex Objects: The Method of Separating Axes*. Elsevier booksite.

Ebert, Frederik et al. (2018). *Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning*. arXiv: 1810.03043 [cs.RO]. URL: https://arxiv.org/abs/1810.03043.

Endres, Felix, Jeff Trinkle and Wolfram Burgard (2013). 'Learning the Dynamics of Doors for Robotic Manipulation'. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3543–3549.

Fang, Irving et al. (2024). 'Fusionsense: Bridging common sense, vision, and touch for robust sparse-view reconstruction'. In: *arXiv preprint arXiv:2410.08282*.

Gadre, Samir Yitzhak, Kiana Ehsani and Shuran Song (May 2021). 'Act the Part: Learning Interaction Strategies for Articulated Object Part Discovery'. In: *arXiv:2105.01047 [cs]*. arXiv: 2105.01047 [cs]. (Visited on 19/04/2022).

Galbally, Elena et al. (2022). 'Elly: A Real-Time Failure Recovery and Data Collection System for Robotic Manipulation'. In: *arXiv preprint arXiv:2208.11845*.

Gammell, Jonathan D and Marlin P Strub (2021). 'Asymptotically Optimal Sampling-Based Motion Planning Methods'. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4, pp. 295–318.

Gavura, Juraj et al. (2025). 'Robotic Calibration Based on Haptic Feedback Improves Sim-to-Real Transfer'. In: *arXiv preprint arXiv:2507.08572*.

Gelfand, Natasha and Leonidas J. Guibas (July 2004). 'Shape Segmentation Using Local Slippage Analysis'. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. SGP '04. New York, NY, USA: Association for Computing Machinery, pp. 214–223. ISBN: 978-3-905673-13-5. DOI: 10.1145/1057432.1057461. (Visited on 02/05/2025).

Goller, Tim et al. (2025). 'Fault Handling in Robotic Manipulation Tasks for Model Predictive Interaction Control'. In: *IEEE Robotics and Automation Letters* 99, pp. 1–8.

Graham, Benjamin, Martin Engelcke and Laurens van der Maaten (June 2018). '3D Semantic Segmentation with Submanifold Sparse Convolutional Networks'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9224–9232. DOI: 10.1109/CVPR.2018.00961. (Visited on 02/05/2025).

Gupta, Arjun et al. (2024). 'Opening cabinets and drawers in the real world using a commodity mobile manipulator'. In: *CoRR*.

Habib, Mohamed Ibrahim (2021). 'Detecting Doors Edges in Diverse Environments for Visually Disabled People'. en. In: *International Journal of Computer Science and Network Security* 21.5, 9–15. DOI: 10.22937/IJCSNS.2021.21.5.2.

Hansen, Johanna et al. (2022). 'Visuotactile-rl: Learning multimodal manipulation policies with deep reinforcement learning'. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8298–8304.

Hsu, David, J-C Latombe and Rajeev Motwani (1997). 'Path Planning in Expansive Configuration Spaces'. In: *Proceedings of International Conference on Robotics and Automation*. Vol. 3. IEEE, pp. 2719–2726.

Huang, Binghao et al. (2024). '3d-vitac: Learning fine-grained manipulation with visuotactile sensing'. In: *arXiv preprint arXiv:2410.24091*.

Huang, Jiahui et al. (June 2021). 'MultiBodySync: Multi-Body Segmentation and Motion Estimation via 3D Scan Synchronization'. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, pp. 7104–7114. ISBN: 978-1-6654-4509-2. DOI: 10.1109/CVPR46437.2021.00703. (Visited on 12/04/2022).

International Federation of Robotics (7th Oct. 2025). *Service Robots See Global Growth Boom*. IFR International Federation of Robotics. URL: https://ifr.org/ifr-press-releases/news/service-robots-see-global-growth-boom (visited on 22/01/2026).

Jain, Advait and Charles C Kemp (2010). 'Pulling Open Doors and Drawers: Coordinating an Omni-Directional Base and a Compliant Arm with Equilibrium Point Control'. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1807–1814.

Jain, Ajinkya et al. (July 2021). *ScrewNet: Category-Independent Articulation Model Estimation From Depth Images Using Screw Theory*. DOI: 10.48550/arXiv.2008.10518. arXiv: 2008.10518 [cs]. (Visited on 02/05/2025).

Jang, Keunwoo, Sanghyun Kim and Jaeheung Park (2023). 'Motion Planning of Mobile Manipulator for Navigation Including Door Traversal'. In: *IEEE Robotics and Automation Letters*.

Jiang, Yongpeng, Yongyi Jia and Xiang Li (2023). 'Contact-aware non-prehensile manipulation for object retrieval in cluttered environments'. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10604–10611.

Jocher, Glenn, Jing Qiu and Ayush Chaurasia (Jan. 2023). *Ultralytics YOLO*. Version 8.0.0. URL: https://github.com/ultralytics/ultralytics.

Kadalagere Sampath, Suhas et al. (2025). 'A Vision-Guided Deep Learning Framework for Dexterous Robotic Grasping Using Gaussian Processes and Transformers'. In: *Applied Sciences* 15.5, p. 2615.

Kalakrishnan, Mrinal et al. (2011a). 'Learning Force Control Policies for Compliant Manipulation'. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4639–4644.

Kalakrishnan, Mrinal et al. (2011b). 'STOMP: Stochastic Trajectory Optimization for Motion Planning'. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4569–4574.

Kang, Gyuree et al. (2024a). 'A Versatile Door Opening System with Mobile Manipulator through Adaptive Position-Force Control and Reinforcement Learning'. In: *Robotics and Autonomous Systems* 180, p. 104760.

— (Oct. 2024b). 'A versatile door opening system with mobile manipulator through adaptive position-force control and reinforcement learning'. In: *Robotics and Autonomous Systems* 180, p. 104760. ISSN: 0921-8890. DOI: 10.1016/j.robot.2024.104760. URL: http://dx.doi.org/10.1016/j.robot.2024.104760.

Karaman, Sertac and Emilio Frazzoli (2011). 'Sampling-Based Algorithms for Optimal Motion Planning'. In: *The International Journal of Robotics Research* 30.7, pp. 846–894.

Karayiannidis, Yiannis et al. (2012). '"Open Sesame!" Adaptive Force/Velocity Control for Opening Unknown Doors'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4040–4047.

Karayiannidis, Yiannis et al. (2016). 'An Adaptive Control Approach for Opening Doors and Drawers under Uncertainties'. In: *IEEE Transactions on Robotics* 32.1, pp. 161–175.

Kavraki, Lydia E et al. (1996). 'Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces'. In: *IEEE Transactions on Robotics and Automation* 12.4, pp. 566–580.

Kawana, Yuki, Yusuke Mukuta and Tatsuya Harada (Oct. 2021). 'Unsupervised Pose-Aware Part Decomposition for 3D Articulated Objects'. In: *arXiv:2110.04411 [cs]*. arXiv: 2110.04411 [cs]. (Visited on 12/04/2022).

Keating, Ryan and Noah J. Cowan (2016). *UR5 Inverse Kinematics*. Course Report M.E.530.646. Johns Hopkins University.

Khoshelham, Kourosh and Sander Oude Elberink (2012). 'Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications'. In: *Sensors* 12.2, pp. 1437–1454. ISSN: 1424-8220. DOI: 10.3390/s120201437. URL: https://www.mdpi.com/1424-8220/12/2/1437.

Kim, Taehyeon et al. (Aug. 2022). 'Improvement of Door Recognition Algorithm Using Lidar and RGB-D Camera for Mobile Manipulator'. In: *2022 IEEE Sensors Applications Symposium (SAS)*, pp. 1–6. DOI: 10.1109/SAS54819.2022.9881249.

Kobilarov, Marin (2012). 'Cross-Entropy Motion Planning'. In: *The International Journal of Robotics Research* 31.7, pp. 855–871.

Kuffner, James J and Steven M LaValle (2000). 'RRT-connect: An Efficient Approach to Single-Query Path Planning'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. IEEE, pp. 995–1001.

Kuribayashi, Yusuke and Kimitoshi Yamazaki (2023). 'Door-Opening Motion Generation Using DAE Modeling for Mobile Manipulators'. In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1–7.

Kwak, Nosan, Hitoshi Arisumi and Kazuhito Yokoi (May 2011). 'Visual Recognition of a Door and Its Knob for a Humanoid Robot'. In: *2011 IEEE International Conference on Robotics and Automation*, pp. 2079–2084. DOI: 10.1109/ICRA.2011.5979608. (Visited on 02/05/2025).

LaValle, Steven M (2006). *Planning Algorithms*. Cambridge university press.

LaValle, Steven M, James J Kuffner, BR Donald et al. (2001). 'Rapidly-Exploring Random Trees: Progress and Prospects'. In: *Algorithmic and computational robotics: new directions* 5, pp. 293–308.

Li, Hao et al. (2016). 'Mobility Fitting Using 4D RANSAC'. In: *Computer Graphics Forum* 35.5, pp. 79–88. ISSN: 1467-8659. DOI: 10.1111/cgf.12965. (Visited on 02/05/2025).

Li, Hao et al. (2022). 'See, hear, and feel: Smart sensory fusion for robotic manipulation'. In: *arXiv preprint arXiv:2212.03858*.

Li, Hongyu et al. (2025). 'ViTa-Zero: Zero-shot visuotactile object 6D pose estimation'. In: *arXiv preprint arXiv:2504.13179*.

Li, Xiaolong et al. (Apr. 2020). 'Category-Level Articulated Object Pose Estimation'. In: *arXiv:1912.11913 [cs]*. arXiv: 1912.11913 [cs]. (Visited on 12/04/2022).

Li, Xing, Manuel Baum and Oliver Brock (Oct. 2023). 'Augmentation Enables One-Shot Generalization in Learning from Demonstration for Contact-Rich Manipulation'. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3656–3663. DOI: 10.1109/IROS55552.2023.10341625. (Visited on 04/05/2025).

Li, Yanbo, Zakary Littlefield and Kostas E Bekris (2016). 'Asymptotically Optimal Sampling-Based Kinodynamic Planning'. In: *The International Journal of Robotics Research* 35.5, pp. 528–564.

Limoyo, Oliver et al. (2018). 'Self-calibration of mobile manipulator kinematic and sensor extrinsic parameters through contact-based interaction'. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4913–4920.

Liu, Qihao et al. (Nov. 2020). *Nothing But Geometric Constraints: A Model-Free Method for Articulated Object Pose Estimation*. DOI: 10.48550/arXiv.2012.00088. arXiv: 2012.00088 [cs]. (Visited on 02/05/2025).

Llopart, Adrian, Ole Ravn and Nils. A. Andersen (Apr. 2017). 'Door and Cabinet Recognition Using Convolutional Neural Nets and Real-Time Method Fusion for Handle Detection and Grasping'. In: *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pp. 144–149. DOI: 10.1109/ICCAR.2017.7942676.

Lu, Weifeng et al. (2025). *RoboFAC: A Comprehensive Framework for Robotic Failure Analysis and Correction*. arXiv: 2505.12224 [cs.RO]. URL: https://arxiv.org/abs/2505.12224.

Meeussen, Wim et al. (2010). 'Autonomous Door Opening and Plugging in with a Personal Robot'. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 729–736.

Meyer Zu Borgsen, Sebastian et al. (2014). 'Automated Door Detection with a 3D-Sensor'. en. In: *2014 Canadian Conference on Computer and Robot Vision*. Montreal, QC: IEEE, pp. 276–282. ISBN: 978-1-4799-4337-1. DOI: 10.1109/CRV.2014.44. URL: https://ieeexplore.ieee.org/document/6816854/.

Mitra, Niloy J. et al. (July 2010). 'Illustrating How Mechanical Assemblies Work'. In: *ACM Trans. Graph.* 29.4, 58:1–58:12. ISSN: 0730-0301. DOI: 10.1145/1778765.1778795. (Visited on 02/05/2025).

Mittal, Mayank et al. (2022). 'Articulated Object Interaction in Unknown Scenes with Whole-Body Mobile Manipulation'. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1647–1654.

Mukadam, Mustafa et al. (2018). 'Continuous-Time Gaussian Process Motion Planning via Probabilistic Inference'. In: *The International Journal of Robotics Research* 37.11, pp. 1319–1340.

Murali, Prajval Kumar, Bernd Porr and Mohsen Kaboli (2025). 'Shared visuo-tactile interactive perception for robust object pose estimation'. In: *The International Journal of Robotics Research* 44.7, pp. 1186–1216.

Nagatani, Keiji and Shinichi Yuta (1996). 'Designing Strategy and Implementation of Mobile Manipulator Control System for Opening Door'. In: *IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE, pp. 2828–2834.

Nemec, Bojan, Leon Žlajpah and Aleš Ude (2017). 'Door Opening by Joining Reinforcement Learning and Intelligent Control'. In: *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, pp. 222–228.

Nonnengießer, Felix et al. (2025). 'In-Hand Object Pose Estimation via Visual-Tactile Fusion'. In: *arXiv preprint arXiv:2506.10787*.

Pan, Jia, Sachin Chitta and Dinesh Manocha (2012). 'FCL: A General Purpose Library for Collision and Proximity Queries'. In: *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866. DOI: 10.1109/ICRA.2012.6225337.

Pan, Tianyang et al. (2022). 'Failure is an option: task and motion planning with failing executions'. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1947–1953.

Pejić, Petra et al. (2023). 'Articulated Objects: From Detection to Manipulation—Survey'. In: *Intelligent Autonomous Systems 17*. Ed. by Ivan Petrovic, Emanuele Menegatti and Ivan Marković. Cham: Springer Nature Switzerland, pp. 495–508. ISBN: 978-3-031-22216-0.

Peterson, L., D. Austin and D. Kragic (2000). 'High-Level Control of a Mobile Manipulator for Door Opening'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3, pp. 2333–2338. DOI: 10.1109/IROS.2000.895316.

Petrović, Luka, Ivan Marković and Ivan Petrović (2022). 'Mixtures of Gaussian Processes for Robot Motion Planning Using Stochastic Trajectory Optimization'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.12, pp. 7378–7390.

Prats, Mario, Pedro J Sanz and Angel P Del Pobil (2010). 'Reliable Non-Prehensile Door Opening through the Combination of Vision, Tactile and Force Feedback'. In: *Autonomous Robots* 29, pp. 201–218.

Qi, Charles R. et al. (June 2017a). *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. DOI: 10.48550/arXiv.1706.02413. arXiv: 1706.02413 [cs]. (Visited on 02/05/2025).

Qi, Charles R. et al. (Apr. 2017b). *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. DOI: 10.48550/arXiv.1612.00593. arXiv: 1612.00593 [cs]. (Visited on 02/05/2025).

Quintana, B. et al. (Jan. 2018). 'Door Detection in 3D Coloured Point Clouds of Indoor Environments'. In: *Automation in Construction* 85, pp. 146–166. ISSN: 09265805. DOI: 10.1016/j.autcon.2017.10.016. (Visited on 04/01/2023).

Racca, Mattia et al. (Oct. 2016). 'Learning In-Contact Control Strategies from Demonstration'. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 688–695. DOI: 10.1109/IROS.2016.7759127. (Visited on 03/05/2025).

Ramôa, João Gaspar, Luís A. Alexandre and S. Mogo (Apr. 2020). 'Real-Time 3D Door Detection and Classification on a Low-Power Device'. In: *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 96–101. DOI: 10.1109/ICARSC49921.2020.9096155.

Ramôa, João Gaspar et al. (2021). 'Real-time 2D–3D Door Detection and State Classification on a Low-power Device'. en. In: *SN Applied Sciences* 3.5, p. 590. ISSN: 2523-3963, 2523-3971. DOI: 10.1007/s42452-021-04588-3.

Ratner, Ellis, Claire J Tomlin and Maxim Likhachev (2023). 'Operating with inaccurate models by integrating control-level discrepancy information into planning'. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7823–7829.

Ren, Kejia et al. (2025). 'Collision-inclusive Manipulation Planning for Occluded Object Grasping via Compliant Robot Motions'. In: *IEEE Robotics and Automation Letters* 99, pp. 1–8.

Rühr, Thomas et al. (2012). 'A Generalized Framework for Opening Doors and Drawers in Kitchen Environments'. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3852–3858.

Salzman, Oren and Dan Halperin (2016). 'Asymptotically Near-Optimal RRT for Fast, High-Quality Motion Planning'. In: *IEEE Transactions on Robotics* 32.3, pp. 473–483.

Schulman, John et al. (2014). 'Motion Planning with Sequential Convex Optimization and Convex Collision Checking'. In: *The International Journal of Robotics Research* 33.9, pp. 1251–1270.

Sermanet, Pierre, Kelvin Xu and Sergey Levine (June 2017). *Unsupervised Perceptual Rewards for Imitation Learning*. DOI: 10.48550/arXiv.1612.06699. arXiv: 1612.06699 [cs]. (Visited on 04/05/2025).

Sferrazza, Carmelo et al. (2024). 'The power of the senses: Generalizable manipulation from vision and touch through masked multimodal learning'. In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 9698–9705.

Sharf, Andrei et al. (2014). 'Mobility-Trees for Indoor Scenes Manipulation'. In: *Computer Graphics Forum* 33.1, pp. 2–14. ISSN: 1467-8659. DOI: 10.1111/cgf.12204. (Visited on 02/05/2025).

Shen, Henghua, Wen-Fang Xie and Ningyu Zhu (2024). 'Degeneracy-Aware Full-Pose Path Planning Strategy for Robot Manipulator'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Šimundić, Valentin, Luka Petrović and Robert Cupec (2026). 'Multi-Contact Robot Arm Path Planning for Opening Handleless Doors'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. DOI: 10.1109/TSMC.2026.3656177.

Šimundić, Valentin et al. (2023). 'Introduction to Door Opening Type Classification Based on Human Demonstration'. In: *Sensors* 23.6. ISSN: 1424-8220. DOI: 10.3390/s23063093.

Šimundić, Valentin et al. (2026). 'Framework for Robot Door Opening Based on Visual, Force, and Tactile Integration'. In: *IEEE Access* 14, pp. 11110–11128. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2026.3655617. (Visited on 27/01/2026).

Skulimowski, Piotr, Mateusz Owczarek and Pawel Strumillo (Sept. 2017). 'Door Detection in Images of 3D Scenes in an Electronic Travel Aid for the Blind'. In: *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, pp. 189–194. DOI: 10.1109/ISPA.2017.8073593.

Stuede, Marvin et al. (2019). 'Door Opening and Traversal with an Industrial Cartesian Impedance Controlled Mobile Robot'. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 966–972.

Sturm, Jürgen et al. (2010). 'Vision-Based Detection for Learning Articulation Models of Cabinet Doors and Drawers in Household Environments'. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 362–368.

Thamrongaphichartkul, Kitti and Supachai Vongbunyong (2024). 'Enhancing Autonomous Door Traversal for Mobile Manipulators Using Quadratic Programming and Cartesian Compliance Control'. In: *IEEE access : practical innovations, open solutions*.

Vats, Shivam, Maxim Likhachev and Oliver Kroemer (2023). *Efficient Recovery Learning using Model Predictive Meta-Reasoning*. arXiv: 2209.13605 [cs.RO]. URL: https://arxiv.org/abs/2209.13605.

Vats, Shivam et al. (2025). *RecoveryChaining: Learning Local Recovery Policies for Robust Manipulation*. arXiv: 2410.13979 [cs.RO]. URL: https://arxiv.org/abs/2410.13979.

Vlaminck, Michiel et al. (Mar. 2016). 'Indoor Assistance for Visually Impaired People Using a RGB-D Camera'. In: *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pp. 161–164. DOI: 10.1109/SSIAI.2016.7459200.

Wang, He et al. (June 2019). *Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation*. DOI: 10.48550/arXiv.1901.02970. arXiv: 1901.02970 [cs]. (Visited on 02/05/2025).

Wang, Jiayu et al. (2020). 'Learning Semantic Keypoint Representations for Door Opening Manipulation'. In: *IEEE Robotics and Automation Letters* 5.4, pp. 6980–6987.

Wang, Zhi et al. (2025). 'DoorBot: Closed-Loop Task Planning and Manipulation for Door Opening in the Wild with Haptic Feedback'. In: *arXiv preprint arXiv:2504.09358*.

Wei, Lai et al. (2025). 'Ensuring Force Safety in Vision-Guided Robotic Manipulation via Implicit Tactile Calibration'. In: *Proceedings of The 9th Conference on Robot Learning*. Ed. by Joseph Lim, Shuran Song and Hae-Won Park. Vol. 305. Proceedings of Machine Learning Research. PMLR, pp. 1049–1062. URL: https://proceedings.mlr.press/v305/wei25a.html.

Welschehold, Tim, Christian Dornhege and Wolfram Burgard (Sept. 2017). 'Learning Mobile Manipulation Actions from Human Demonstrations'. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3196–3201. DOI: 10.1109/IROS.2017.8206152.

Xu, Weiwei et al. (July 2009). 'Joint-Aware Manipulation of Deformable Models'. In: *ACM Trans. Graph.* 28.3, 35:1–35:9. ISSN: 0730-0301. DOI: 10.1145/1531326.1531341. (Visited on 02/05/2025).

Yi, Li et al. (Dec. 2018). 'Deep Part Induction from Articulated Object Pairs'. In: *ACM Transactions on Graphics* 37.6, pp. 1–15. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3272127.3275027. arXiv: 1809.07417 [cs]. (Visited on 01/10/2025).

Yuan, Qing et al. (2016). 'Space-Time Co-Segmentation of Articulated Point Cloud Sequences'. In: *Computer Graphics Forum* 35.2, pp. 419–429. ISSN: 1467-8659. DOI: 10.1111/cgf.12843. (Visited on 02/05/2025).

Zeng, Vicky et al. (Dec. 2021). *Visual Identification of Articulated Object Parts*. DOI: 10.48550/arXiv.2012.00284. arXiv: 2012.00284 [cs]. (Visited on 02/05/2025).

Zhang, Ge et al. (May 2021). *StrobeNet: Category-Level Multiview Reconstruction of Articulated Objects*. DOI: 10.48550/arXiv.2105.08016. arXiv: 2105.08016 [cs]. (Visited on 02/05/2025).

Zucker, Matt et al. (2013). 'Chomp: Covariant Hamiltonian Optimization for Motion Planning'. In: *The International journal of robotics research* 32.9-10, pp. 1164–1193.