ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Master Thesis

Interaction of a Mobile Manipulator With Articulated Objects

Spring Term 2021

Supervised by:
Michel Breyer
Giuseppe Rizzi
Julian Förster

Author:
Marko Maljkovic

Contents

Acknowledgements

I would like to thank my supervisors, Michel Breyer, Giuseppe Rizzi and Julian Förster for their advice, patience and support throughout the course of the project.

Abstract

Nowadays, many robots are finding their ways into our everyday lives. Our ultimate goal would be to increase their overall utility by having them perform manipulation tasks robustly, safely and in real time in environments designed for humans. Hence, one of the first tasks they would have to learn is how to open different types of doors such as room doors, sliding doors, drawers, containers and lids. In essence, this is a multidisciplinary task that comprises of machine perception, trajectory planning and whole body control. In this work, we focus on the last two aspects in order to execute a door opening procedure once the handle has been grasped. Since this is a manipulation task, we have to circumvent the absence of a reliable algebraic model of the continuously changing contact forces that could be used with the traditional, model based, control approaches. We leverage the fixed-grasp assumption and build our system on top of that. Traditional approaches heavily rely on cameras but in this work, an approach is presented that relies solely on the combination of the haptic and proprioceptive feedback available through force/torque sensors and joint encoders. More importantly, it requires no a priori information about the mechanism that is operated. We propose to use a three-step cyclic procedure that starts with fitting the constraint model to the operated mechanism, then moving one step along the planned trajectory and finally, closing the loop by measuring the robot's state and refitting the model for future use. We first design and validate our approach in simulation by testing three different mobile base problem formulations that maximize robot's manipulability during the run. After comparing their performance to a fixed base algorithm and choosing the optimal problem formulation, an extensive analysis of the model's performance is reported. We then do a hyperparameter exploration procedure and report the most promising combination of the parameters for future use on the real robot. In order to test the low level control, we first adapt the algorithm for the Gazebo simulation, where we demonstrate the ability of the robot to follow simple commanded trajectories in the world frame. At the end, through extensive experiments on the real robot, we show that the proposed system is indeed capable of opening a drawer and a cabinet door mechanism.

Symbols

Indices

x	x axis
y	y axis

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
RL	Reinforcement Learning
DOF	Degree of freedom
RANSAC	Random sample consensus
PRM	Probabilistic road map
RRT	Rapidly exploring random trees
MPC	Model predictive control
MLESAC	Maximum likelihood RANSAC
URDF	Unified robot description format
IK	Inverse kinematics
ROS	Robotic operating system
PCA	Principal component analysis
SVD	Singular value decomposition
QCCO	Quadratically constrained convex optimization
LCQP	Linearly constrained quadratic program
SOCP	Second order cone program

Chapter 1

Introduction

1.1 Motivation

Object manipulation has over the years become one of the most studied problems in robotics. Since we strive to deploy robots in environments designed for humans, we need to design them such that they can reliably and in real time operate articulated objects, in particular room doors, sliding doors, drawers and lids. The range of applications (robot assistants, service robots, collaborative robots etc.) and the overall utility of a robot which is to navigate indoors is dictated by its ability to open doors. This still remains an open problem in the general case, despite the increase in attention it has received in the last decade. The complete procedure can be split into three steps:

1. Identifying the position, shape and size of the door and the handle
2. Approaching and grasping the handle
3. Sensor-based planning and coordinated arm-base execution of the door opening trajectory

To complete the first step, one usually turns to either image segmentation and classification techniques [1, 2, 3, 4] or extracting information from laser scans (lidars) [5]. In addition to traditional methods of extracting shapes from point clouds (such as RANSAC [6], line regression etc.), the advent of machine learning, and especially deep learning, has allowed researchers to choose between a plethora of pre-trained, already published models that can be easily plugged in and used jointly with the remaining part of the algorithm. On the other hand, the main issue that the second step deals with is the problem of precise navigation inside the room such that a desired handle grasp can be achieved. Since the main focus of this thesis revolves around the closed-loop whole body control, in this project we will address the third step only. Namely, we assume that upon encountering a new type of door, some external algorithm allows the robot to successfully approach and grasp the handle, leaving it to us to perform the door opening procedure.

In essence, there are three key aspects we have to pay attention to when designing the door opening procedure:

- As mobile manipulators make their way into many application areas, the range of different doors a robot can encounter throughout its lifetime is very wide. Since doors vary in shape, size, joint mechanism etc., a priori computing the door opening motion for every combination is infeasible. Therefore, the robot must be able to autonomously adapt to different types of mechanisms.

- For the system to be adaptive, it must also be able to locally perceive the environment using the available on-board sensors. Nowadays, typically used sensors include encoders, force/torque sensors, lidars and cameras. Based on the robot application, one should carefully design the perception module such that a trade-off between the richness of the extracted information and the computational time required to extract the information is achieved.
- Since in most cases we work with manipulators on a mobile base, in order to be flexible regarding the spatial extent of the mechanism to be operated, the opening trajectory should be carried out using a whole-body controller such that the degrees of freedom provided by the base are also used. Computing such a motion can be challenging as it requires precise coordination between the movement of the arm and the base such that desired trajectory of the end effector is achieved.

With these three aspects in mind, one should be able to use online measurements to fit different kinematic models of the operated mechanisms in real time, and in return, help improve the appropriate door opening trajectory. Due to available hardware, in this thesis we consider a system which has a robotic arm with 7 degrees of freedom (DOF) mounted on an omnidirectional mobile base with encoders and force/torque sensors mounted at each joint at our disposal.

1.2 Related Work

Successful applications in the literature generally tackle the planning and control problems in four different ways:

1. Reinforcement learning (RL)
2. Sampling based approach
3. Optimization based approach
4. Other traditional feedback based control approaches that do not utilize optimization, sampling or RL

The first group of algorithms is known for its capability to learn complex tasks. However, making a RL agent generalize over different instances of doors and drawers is very challenging as it would either require training multiple agents in different setups and designing an agent-switching policy to be deployed when executing the task, or learning a universal policy through an extensive domain randomization during training. In [7], a Q-learning approach is utilized to learn the relational representation of a kinematic structure to be manipulated. Though this work was successful in learning the kinematic chain of several complex articulated objects, the procedure required camera sensors, image processing and extensive interaction with the object, making it computationally too expensive for our real time application. To help achieve training times that are practical for real physical systems, [8] utilizes multiple experience collector threads and one training thread. Though the authors test the algorithm using two real robots, they only present the setup and the results for one type of door. On the other hand, [4, 9] propose using reinforcement learning in a framework where learning the policy is governed by some traditional control approach. The traditional method is used to simultaneously execute the task at hand and collect the data for the RL agent. When it is no longer feasible to use the traditional method, the robot utilizes the control policy provided by the RL agent. The approach presented in [4] is not directly applicable in our scenario as it focuses on a block assembly task and also relies on camera tracking. However, it can serve

as an inspiration for designing the reward functions and the agent's architecture as it addresses the problem of actuator constraints by penalising high values of the actuator commands through smaller values of the corresponding reward. The work done in [9] is focused on the door opening task and presents a successful RL implementation governed by impedance control, though the experimental results were presented only for one type of the door. In [10, 11], authors leverage the idea of a universal policy for different parametric models, with the latter paper introducing online system identification framework. These approaches rely on collecting huge amounts of training experience which would, in our case, be infeasible when using the real robot. Finally, none of these papers address the scenario in which the robot base is mobile.

The second group of algorithms samples from the configuration space of the robot to usually build upon Probabilistic Road Map (PRM) [12] or Rapidly-exploring Random Trees (RRT) [13] motion planning approaches. The robot's configuration space is discretized, sampled according to a heuristic that should qualitatively describe the task at hand and evaluated against the constraints imposed by the system. If the constraints are satisfied, the prospective sampled configuration is accepted and transferred to a reference tracking control module. Examples of such works are presented in [2, 14, 15]. Though these approaches can tackle highly nonlinear and constrained problems, they suffer from the huge state dimensionality after discretization and having to rely on a precise complete model of the system required for the evaluation of the constraints. This model has to be either a priori known or extracted from the visual input, neither of which is considered plausible in our framework. Furthermore, the constraint evaluation step is usually very computationally expensive thus making this kind of approach less suitable for our application. None of the mentioned papers consider the mobile base scenario.

Optimization based approaches allow for optimal planning within feasible set of decision variables imposed by the hardware and task constraints. For each particular problem, an objective function is defined such that the value that minimizes the objective corresponds to the optimal inputs to the system. If the objective function is convex, the global optimality of the solution is guaranteed. This is particularly suitable for satisfying the joint position, velocity and acceleration constraints. Though there are commercially available, non-convex solvers, most of the time-efficient ones are gradient based, iterative procedures, best suited for convex optimization problems. The most common approach in the literature is model predictive control (MPC). Some works such as [16, 17, 18] utilize MPC to achieve sensor based, task constrained, whole body control for reference tracking. In order to use MPC for trajectory planning, one would have to develop an algebraic model of the external forces acting on the robot. If we want this model to adequately describe the continuously changing interaction forces between the object and the end effector, it would have to be learnt online, while the door opening procedure is executed. Given that at each iteration, an optimization problem (with potentially high number of optimization variables) would have to be solved, combining MPC approach with a precise model identification procedure could yield a violation of the real time requirements. On the other hand, optimization is also used for hierarchical null space projection control as described in [19, 20, 21]. Though these works do not address the door opening problem, they serve as a good insight into how different mobile base problems can be formulated in accordance with the null space projection control method. In all of the papers, tasks to be solved are divided into groups of a certain priority such that each task is always solved in the joint null space of all the tasks that have a higher priority. Usually, only one-step-ahead optimization is performed, which yields shorter average times required for each iteration compared

to the MPC approach making it a more suitable approach for our problem setup.

All other approaches fall under the category of traditional feedback control. In [3, 1], a compliant control of the arm is combined with velocity control of the base. The former paper is based on the fixed-grasp assumption, meaning that the door handle position can always uniquely be inferred from the position of the end effector. Parametric models for the prismatic and revolute joints are fitted at each iteration, and based on the one that better fits the end effector motion, the door opening trajectory is adapted. Though easy to compute, the least squares models are vulnerable to noise and relative motion between the end effector and the handle, making them an unreliable choice for the real world application. The latter approach, just like [22], alleviates this problem by using a probabilistic framework for model selection in combination with a more robust, Maximum Likelihood RANSAC method (MLESAC) [23]. In [24], a successful implementation of Equilibrium Point Control is presented. It is combined with a simple mobile base control strategy and successfully tested on different types of drawers and cabinet doors with revolute joints. However, the presented work relies on an unconventional end effector type in a form of a hook which reduces the overall utility of the proposed algorithm and does not comply with the fixed-grasp assumption. Finally, in [25], a concept of control in the direction of maximal mobility is introduced. The manipulability index is used to determine how much the end effector can be operated freely and was first introduced in [26]. The concept of maximal manipulability is important in our problem setup as the door opening procedure is usually just one subtask in a larger manipulation sequence. Thus, we want to maximize the manipulability of the arm during the door opening procedure in order to avoid moving into configurations from which the remaining part of the task is no longer feasible.

1.3 Project Goals

Most types of doors operate using prismatic or revolute joint mechanisms. Due to computational efficiency and the capability to perform planning within constraints, our method is best described as a combination of the optimization based approach and the traditional feedback control, whose main goal is to autonomously open different types of doors. To achieve so, we organize the main task into four different sub-goals, each referring to a successful implementation of a particular module included in the complete closed loop control algorithm. The schematic representation of the four-stage, one step look ahead approach that we take is given in Figure 1.1.

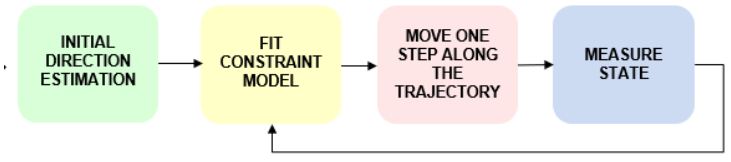


Figure 1.1: Schematic representation of the door opening procedure.

As depicted, we organise our setup in three main modules and one additional, active only at the beginning of the procedure:

Initial direction estimation module: This module is activated only at the start of the procedure. It is used to give an initial estimate of the direction in which

the robot should pull the door open. However, its purpose is not to provide the perfect estimate, but rather to be good enough to achieve a slight movement in the right direction such that an initial value can be provided to the online direction estimation module. Upon grasping a new door handle, due to no external information about the relative orientation between the end effector and the handle, robot cannot a priori decide in which direction to start pulling. Instead, it should try out different directions and based on the received measurements decide what is the best direction to take.

Online direction estimation module: At each time step, based on the measurements obtained from the joint encoders and force/torque sensors mounted at each of the joints, the online direction estimation module re-calculates the estimate of the direction in which the end effector can move freely.

End effector velocity planner: After the new direction estimate is obtained, it is fed to the end effector velocity planner module. Angular velocity is planned to align one of the axes of the end effector frame with the door's normal vector. A specific temporal profile of the end effector's linear velocity profile is used to ensure that the total linear velocity of the end effector is smaller during the short initial period where the estimate has not yet converged. Furthermore, to utilize the additional DOFs of the mobile base, we split the total desired linear velocity into the arm and mobile base components. We do this through an optimization procedure such that we maximize the arm's manipulability index in order to better prepare the robot for the tasks that follow. Finally, this module performs convex optimization to find the optimal joint velocities within hardware constraints.

Observation module: This module is the final requirement for closing the loop. Joint encoders provide us with the values of joint positions, velocities and applied torques. Measurements from the force/torque sensors mounted at each joint are internally combined such that an estimate of the external wrench acting on the end effector frame is obtained. Finally, through vendor specific libraries for parsing the URDF (Unified Robot Description Format) model of the robot, we receive the information about the mass matrix, gravity vector, Coriolis forces and end effector jacobian for a particular joint state.

With everything previously said in mind, the complete list of sub-goals that we try to achieve in this thesis can be formulated as follows:

- Design and implementation of a robust initial direction estimation module that performs fast, simple, jerk movements from which it is able to infer a direction which would yield at least a slight movement in the appropriate direction.
- Under the fixed grasp assumption, we want to design a fast, adaptive online direction estimator using the estimated external wrench feedback. As we are assuming a fixed-grasp model, it is important to avoid contact point switching.
- Design and implementation of the velocity planner module in charge of issuing the control commands for both the mobile base and the robotic arm.
- Closing the loop and testing the whole body control algorithm in a simulated physics environment available through PyBullet [27] library.
- Analyzing the average planning time per iteration, manipulability index and convergence of the direction estimator to pick the best combination of parameters and optimization solvers.

- Integrating the velocity planner implementation with ROS [28] framework, writing the custom controller plugin and testing the communication and control pipeline in Gazebo [29] simulation.
- Testing the algorithm on prismatic (drawer) and revolute (cabinet door) joint mechanisms using a real Franka Emika's research robot mounted on a Ridgeback mobile base.

1.4 Thesis Organization

This thesis consists of 9 chapters. Chapter 2 is a short overview of the method, simulation and robot setups that we use throughout the thesis. In Chapter 3, we present the initial direction estimator. Chapters 4 and 5 thoroughly describe the online direction estimator and velocity planner modules. For the velocity planner module, we present several appropriate formulations of the problem in order to facilitate several commercially available optimization solvers. Chapter 6 addresses the final implementation aspects in the Gazebo simulation and on the real robot. Chapter 7 is devoted to results obtained in simulated environments. We compare different problem formulations introduced in Chapter 5, perform hyper-parameter exploration, show the performance of the system when initial direction estimation is included and test the low level control in the Gazebo simulation. Finally, we present the results obtained on the real robot in Chapter 8 and summarize our conclusions and outline future work in Chapter 9

Chapter 2

Overview of the Method, Simulations and Robot Description

To begin the algorithm design process, we first opt for extensive research and testing in simulation. This chapter serves as a detailed description of the method that the robot will apply, the tasks and mechanisms the robot will encounter, the robot models used in each of the simulations and the actual hardware used for the real world testing. We perform two types of simulations:

1. **Pybullet simulation:** This simulation is used as a proof of concept that our four modules combined together successfully perform the door opening procedure. It will cover 5 different door models.
2. **Gazebo simulation:** The main purpose of this simulation is to organize and test the algorithm developed in PyBullet simulation in a ROS Control framework. The code organization in this simulation closely resembles the one that will be used on the real robot.

2.1 Overview of the Whole Body Control Method

In essence, our method can be split into 4 large modules, as previously introduced in Figure 1.1, whose functionalities have to be analyzed. The initial direction estimation module is active only at the beginning of the whole body control procedure whereas the remaining three modules form an iterative feedback control procedure. Figure 2.1 shows the overview of all sub-modules that are used in our approach and the way they are interconnected.

We a priori set the total number of loop iterations, denoted as N_{total}. Furthermore, inside the loop we define the 'start-up' period of the iterative part of the procedure in which we assume that the unconstrained direction of motion estimate has not yet converged. The 'start-up' period covers the first:

$$N_{\text{start-up}} = \left\lfloor \frac{N_{\text{total}}}{10} \right\rfloor \tag{2.1}$$

loop iterations and will enforce smaller magnitude of the end effector's linear velocity. By construction, the velocity planner (marked in red on the picture) consists of most components and accounts for most of the execution time per loop iteration.

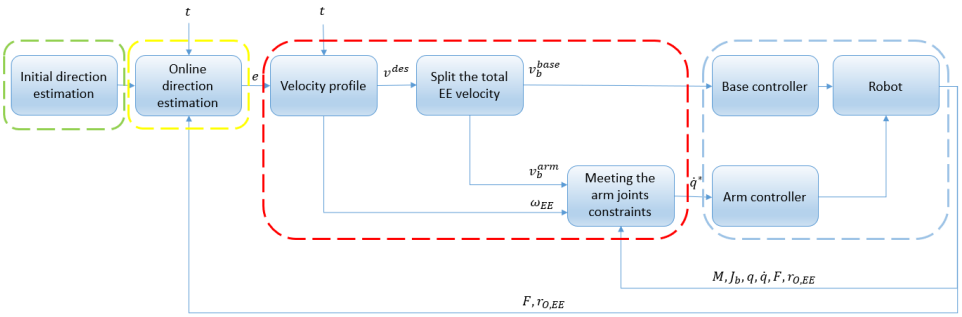
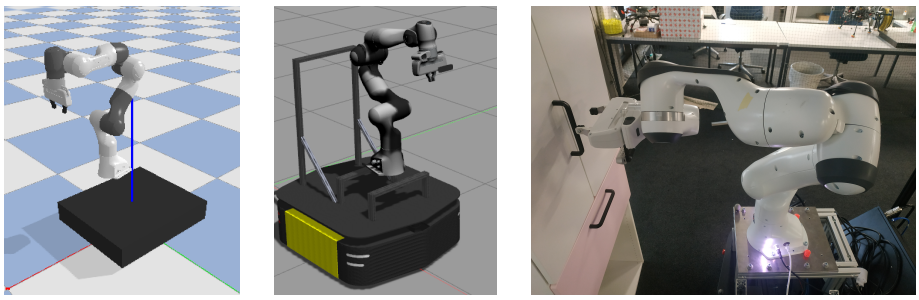


Figure 2.1: Block diagram of the proposed method.

For the fixed base scenario of the algorithm (that will also be analysed in this thesis), the 'Split the total EE velocity' sub-module is simply an identity transformation. All other components remain the same.

2.2 Robot

The model of the robot that we use in both simulations corresponds in design exactly to the real hardware that we have available. It consists of a robotic arm from the Franka Emika manufacturer [30] and the omnidirectional mobile base from Clearpath Robotics [31]. Figure 2.2 shows the two models used in the simulation and the real hardware setup.



(a) Robot in PyBullet (b) Robot in Gazebo (c) Real world robot

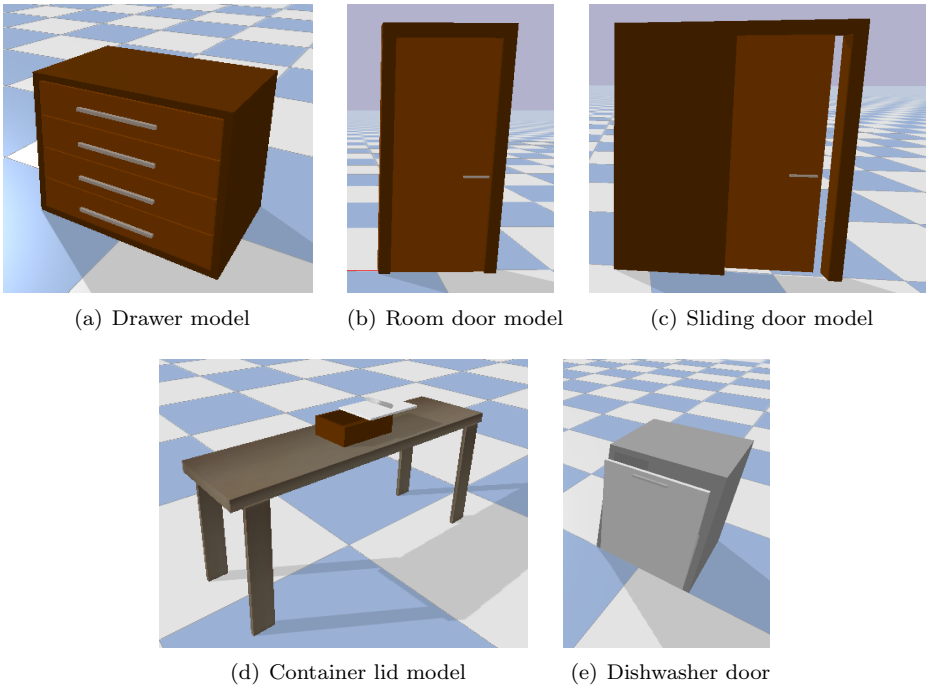
Figure 2.2: 'RoyalPanda' robot used in simulations and real world experiments.

In total, the robot has 10 DOFs: 7 from the robotic arm and additional 3 from the mobile base. We denote the arm's joint positions as $q = [q_1, q_2, \ \dots \ , q_7]^T$, joint velocities as $\dot{q} = [\dot{q}_1, \dot{q}_2, \ \dots \ , \dot{q}_7]^T$ and joint accelerations as $\ddot{q} = [\ddot{q}_1, \ddot{q}_1, \ \dots \ , \ddot{q}_7]^T$. The arm can be controlled by commanding desired joint positions, joint velocities, joint torques or the Cartesian end effector linear and rotational velocities. In this project, we will be calculating desired joint velocity commands and utilizing internal joint impedance controller of the arm. On the other hand, the omnidirectional base is controlled by issuing desired linear velocities in the x and y directions of the robot's body frame and the rotational velocity around the z axis.

2.3 Simulation

The PyBullet simulation is organized in *skills* that our robot model can perform. The particular skill that we are developing is called ***door opening skill***. We rely on previously developed skills such as: ***navigate*** and ***grasp***. The first one places the robot in the environment such that the door handle is within robot's reach. The second one, on the other hand, solves inverse kinematic (IK) problem and grasps the handle with a desired relative orientation. As these two skills correspond to the steps 1 and 2 described in Section 1.1, their implementation aspects are beyond the scope of this thesis and will not be further discussed.

The base classes of the direction estimation and the velocity planner modules provide a set of parameters and procedure declarations to be implemented such that any new developed estimator or velocity planner can be used in 'plug and play' fashion. For testing the complete closed loop control, we provide different door models based on prismatic and revolute joint mechanisms:



(a) Drawer model (b) Room door model (c) Sliding door model

(d) Container lid model (e) Dishwasher door

Figure 2.3: Different door models used in PyBulelt simulation.

- **Prismatic joints:** drawer, sliding door and sliding container lid
- **Revolute joints:** standard room door, dishwasher door

As presented in Figure 2.3, a total of 5 URDF door models are used to test the algorithm. The simulation step for all the tests is set to a default value of $T_s = 1/240\,\mathrm{s}$. It is important to note that the PyBullet simulation is executed in a sequential manner. This means that while the commands are being computed, the simulation holds still. Only after the computation is finished, the physics simulation is executed for T_s seconds with the commands fixed at all times. Such a scenario does not correspond to the real world so to test the asynchronous execution, we develop a Gazebo simulation.

2.4 ROS Integration

As previously mentioned, the Gazebo simulation is used primarily to test the ROS communication between different components required to provide the real-time capability to the system that will run on the real robot. We test the communication between:

- The ROS node that integrates the online direction estimator and velocity planner modules into a state machine.
- The ROS node with corresponding services that read the state and control the gripper.
- The ROS node with corresponding service that reads the current state of the robot joints and uses the **libfranka** [30] library to calculate the mass matrix, gravity vector, Coriolis forces and jacobian of the end effector.
- The ROS control plugin that sends commands to the Franka arm.
- The ROS control plugin that sends commands to the Ridgeback mobile base.

Unfortunately, Gazebo is deemed unreliable for the manipulation task simulations. It requires significant tuning to enable simple grasping tasks to be executed due to problems with friction simulation. Therefore, in this segment we do not test the force feedback closed loop control (and do not need to include the door models in the simulation). Instead, we test the capability of the system to follow predefined trajectories in the Cartesian workspace.

An important issue that arose from the Gazebo simulation is the fact that the ROS control plugins have to send commands at a fixed frequency of $f = 1\,\text{kHz}$. This high of a frequency could cause a problem if we wish to plan the desired velocities within the time interval between the two consecutive update steps of the controller. In reality, we do not need to re-plan the trajectory at such a high frequency. Instead, we update the commanded reference values to the control plugins whenever the optimization procedure is completed and in the meantime use the fixed values from the previous optimization iteration. From this point on, whenever we refer to the sampling interval, we consider a time interval sufficient for all the planning steps within one iteration of our algorithm and not the 1 kHz update frequency of the ROS controller plugin.

In the following chapters, we will give a detailed theoretical description of all the sub-modules, starting with the initial direction estimation block.

Chapter 3

Initial Direction Estimation

Upon grasping a door handle, without some external sensors that tell us the exact pose of the door, it is impossible to determine the exact relative orientation between the end effector frame and the door handle frame. As previously mentioned in Section 1.2, a camera system could extract the normal vector of the door plane. Unfortunately, even if we have the information about the door normal, we cannot a priori be sure in which direction to start pulling as this door could be anything from a sliding door opened by pulling to the left or right, to a drawer opened by pulling exactly in the direction of the door normal vector. In order to disambiguate the initial pulling direction, we have to shortly interact with the door mechanism and infer an initial unconstrained direction suitable for moving the end effector of the robot. Using random initialization of the online direction estimation procedure could cause high values of the interaction forces between the end effector and the operated object. This usually leads to a termination of the procedure due to safety reasons, if the online estimation does not compensate for the poor initialization fast enough. In addition, inadequate initialization is more likely to cause the end effector to slip off the handle. We propose to use only the force sensor feedback and a set of very short movements of the end effector in different directions. Our method is based on the fixed-grasp assumption and the assumption that the movement in the unconstrained direction should yield smaller magnitudes of the estimated external wrench acting at the end effector frame.

3.1 Choosing Candidate Initial Directions

The initial direction candidates are samples from the unit half-sphere centered at the grasping point. We consider only the candidates that form an angle smaller than $\pi/2$ rad with the $-z$ axis of the end effector frame, as shown in Figure 3.1(a). To obtain the three dimensional vector, we first discretize the x-y plane of the end effector frame by specifying a desired resolution in both directions. If we set N_x and N_y to be odd numbers of equidistant potential x and y coordinates from the interval $[-1, 1]$, then the corresponding resolutions along each of the axis are given by:

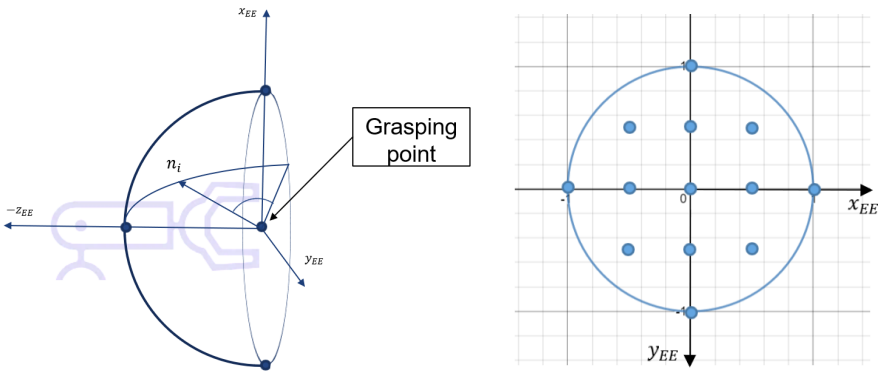
$$\Delta x = \frac{2}{N_x - 1} \text{ and } \Delta y = \frac{2}{N_y - 1} \tag{3.1}$$

The set of all ordered pairs (x_i, y_j) that fall within the unit circle determines the set of all initial direction candidates D. Figure 3.1(b) shows the set of all ordered pairs for the case $N_x = 5$ and $N_y = 5$. Based on the previously determined set of

all ordered pairs, the z components of the candidate vectors are determined such that the Euclidean norm of the vectors is equal to 1. The set D is given by:

$$D = \left\{ \begin{bmatrix} x_i \\ y_j \\ -\sqrt{1 - x_i^2 - y_j^2} \end{bmatrix} : x_i = i\Delta x \wedge y_j = j\Delta y \wedge x_i^2 + y_j^2 \leq 1 \right\} \tag{3.2}$$

where $i \in \left\{-\frac{N_x-1}{2}, \ldots, 0, \ldots, \frac{N_x-1}{2}\right\} \wedge j \in \left\{-\frac{N_y-1}{2}, \ldots, 0, \ldots, \frac{N_y-1}{2}\right\}$.



(a) Sampling vectors on the unit sphere (b) Chosen points for $N_x = 5$ and $N_y = 5$

Figure 3.1: Schematic representation of choosing the initial direction candidates.

3.2 Calculating the Initial Direction

Having defined the set of candidate initial directions, let us introduce the following notation needed for calculating the initial direction estimate:

- V : the small, fixed magnitude of the end effector's linear velocity commanded during the movement in each of the candidate directions.
- $C_{b,EE}$: rotation matrix that maps from the end effector frame to the robot's base frame.
- $J^{\text{lin}}_{b,EE}$: the Jacobian that maps joint velocities to the end effector's linear velocity expressed in the robot's base frame.
- N_{m} : time length of each movement expressed as the number of sampling intervals during which we keep a fixed end effector linear velocity in the direction corresponding to the current candidate.

During initial direction estimation we keep the base fixed at all times. The complete pipeline of the initial direction estimation is presented in **Algorithm 1**. For every candidate direction d, before we start performing the movement, we measure the magnitude of the estimated external force acting on the end effector frame, denoted as $\left|\vec{F}_{\text{start}}\right|$. After recording the measurement, we command the joint velocities that should move the end effector with the linear velocity $v^{lin}_{b,EE} = Vd$ (expressed in the base frame) for N_{m} sampling intervals (the OptimalValuesWithinConstraints procedure in Algorithm 1 respects the joint position, velocity, acceleration and torque constraints and will be explained in detail in Section 5). Upon completion of the movement, we once again measure the magnitude of the external force acting on

Algorithm 1: Initial direction estimation

input : D, V, N_m
output: Initial direction d_init

$N_d = \mathrm{NumberOfElements}(D)$;
for $i \leftarrow 1$ **to** N_d **do**
  $d \leftarrow D(i)$;
  $\vec{F}_{start} \leftarrow \mathrm{ReadExternalForce}()$;
  for $j \leftarrow 1$ **to** N_m **do**
    $J^{\mathrm{lin}}_{b,EE} \leftarrow \mathrm{ReadJacobian}()$;
    $C_{b,EE} \leftarrow \mathrm{ReadEEPose}()$;
    $\dot{q} = \mathrm{OptimalValuesWithinConstraints}(J^{\mathrm{lin}}_{b,EE}, C_{b,EE}, Vd)$;
    $\mathrm{SendJointCommands}(\dot{q})$;
  $\vec{F}_{end} \leftarrow \mathrm{ReadExternalForce}()$;
  $w_i = \max\left\{0, 1 - \frac{|\vec{F}_{end}|}{|\vec{F}_{start}|}\right\}$;
  for $j \leftarrow 1$ **to** N_m **do**
    $J^{\mathrm{lin}}_{b,EE} \leftarrow \mathrm{ReadJacobian}()$;
    $C_{b,EE} \leftarrow \mathrm{ReadEEPose}()$;
    $\dot{q} = \mathrm{OptimalValuesWithinConstraints}(J^{\mathrm{lin}}_{b,EE}, C_{b,EE}, -Vd)$;
    $\mathrm{SendJointCommands}(\dot{q})$;
for $i \leftarrow 1$ **to** N_d **do**
  $p_i = \frac{w_i}{\sum_i w_i}$
$d_\mathrm{init} = \mathrm{Projection}(\mathbb{E}_{x\sim p}[x])$;

the end effector frame, now denoted as $\left|\vec{F}_{\mathrm{end}}\right|$. As previously mentioned, we assume that the movements in the directions close to the actual unconstrained direction of motion should yield $\left|\vec{F}_{\mathrm{end}}\right| \leq \left|\vec{F}_{\mathrm{start}}\right|$. Thus, we assign to each of the candidate directions a corresponding weight given by:

$$w = \max\left\{0, 1 - \frac{\left|\vec{F}_{\mathrm{end}}\right|}{\left|\vec{F}_{\mathrm{start}}\right|}\right\} \tag{3.3}$$

By construction, this weight assignment procedure favours directions that cause the magnitude of the external force to decrease. The candidate directions that cause bigger relative decrease in the external force magnitude will have a bigger weight assigned which complies with our assumption. On the other hand, the directions that cause the magnitude to increase will be deemed more constrained and will be assigned a weight 0 (since in that case $1 - \frac{\left|\vec{F}_{\mathrm{end}}\right|}{\left|\vec{F}_{\mathrm{start}}\right|} < 0$).

By normalizing the weights as:

$$p_i = \frac{w_i}{\sum_i w_i} \tag{3.4}$$

we define a probability distribution over the set of ordered pairs (x_i, y_j) that induced the set D. We now calculate the expected x and y coordinates of the unconstrained direction as:

$$\left(x_i^*, y_j^*\right) = \mathbb{E}_{d=(x_i,y_j)\sim P}[d] \tag{3.5}$$

and the expected unconstrained direction of motion as:

$$d^* = \begin{bmatrix} x_i^* \\ y_j^* \\ -\sqrt{1 - x_i^{*2} - y_j^{*2}} \end{bmatrix} \tag{3.6}$$

Finally, to obtain the initial direction estimate that is to be fed to the direction estimation module, we take the projection of the expected unconstrained direction of motion on the plane perpendicular to the axis along which the gravity acts. This is based on the following assumption:

- Regardless of the door type, the initial pull direction should always be either the direction in which the gravity acts or in a plane parallel to the ground plane.

This assumption holds for most traditional door mechanisms, whose door plane (when the door is closed) is either parallel to the ground or perpendicular to it. If we expect to encounter an unconventional type of door, then we should directly feed d^* to the online direction estimation module. The case when the initial direction should be aligned with the gravity direction corresponds to the case of unattached lids (free floating joint). These cases can be solved with simple movements upwards and as such are not considered in this project. Therefore, we focus on the mechanisms that require the initial pulling direction to be in the plane perpendicular to the gravity vector.

For a plane whose normal vector is n, the projection operator is given by:

$$\bar{P}(n) = \mathbb{I} - nn^T \tag{3.7}$$

In our setup, gravity acts along the z axis of the body frame so the initial unconstrained direction of motion estimate is given by:

$$d_{\text{init}} = \bar{P}\left(C_{EE,b} \cdot z_{b,EE}\right) \begin{bmatrix} x_i^* \\ y_j^* \\ -\sqrt{1 - x_i^{*2} - y_j^{*2}} \end{bmatrix} \tag{3.8}$$

This value of initial direction estimate is effectively used to initialize the state estimation procedure performed in the online direction estimation module of the iterative part of the door opening procedure.

Chapter 4

Online Direction Estimation

Now that we have defined the initial direction estimate, let us define the remaining inputs of the online direction estimation module:

- k : current sampling interval measured from the start of the iterative part of the door opening procedure.
- $\vec{F}_{EE}^{k}$: current measurement of the external force acting on the end effector expressed in the end effector frame.
- e_{EE}^{k} : estimate of the unconstrained direction of motion before the update in the current iteration takes place. For $k = 0$ we set $e_{EE}^{0} = d_{\text{init}}$.
- $\mathcal{L}_r$: a buffer of the last L_{buff} end effector positions expressed in the world frame.

The task of this module is to calculate the updated value of the estimated unconstrained direction, denoted as e_{EE}^{k+1}. This value is later given as one of the inputs to the velocity planner module. We have to pay closer attention to two main issues that arise when designing the online direction estimation procedure:

1. **Relative motion between the end effector and the door handle:** It directly violates the fixed-grasp assumption. Though we try to prevent this from happening by using high values of grasping force, we need to also design a robust solution capable of correctly estimating the unconstrained direction of motion even if some slipping occurs.
2. **Contact point switching:** Given that we constantly update the direction in which we pull the door open and the fact that we use high values of grasping force, it is to be expected that our external wrench estimate suffers from rapid changes in force direction.

To alleviate these problems, we organize our online direction estimation approach in three sub-modules: haptic based estimator, fixed-grasp estimator and a second order Butterworth filter. During the start-up period of the iterative procedure, only the force feedback estimator is active as we are in this time period still collecting the end effector position data required for the estimate based on the fixed-grasp assumption. After the start-up period is completed, the two independently obtained estimates are combined into one that is filtered and transferred to the velocity planner module. In the following three sections we will describe the three aforementioned sub-modules.

4.1 Haptic Based Estimator

The force feedback estimate is used throughout the entire iterative part of the door opening procedure and is our only estimator during the start-up period. It should detect a change in resistance exerted by the environment and adapt the motion direction accordingly. A similar idea to the one presented in Section 3 is used. At the beginning of each iteration, we measure the external force acting on the end effector and argue that if we have a good estimate, the projection of this measurement onto a plane perpendicular to the vector of the current estimate should be very small. If the estimate is good, then the environment surrounding the door mechanism should generate little resistance in the direction perpendicular to the direction of motion. We use this to iteratively update the estimate by trying to get this projection to be as close as possible to 0. Important to note is that we use the same reasoning as in Section 3.2 and take into consideration only the external force projected on the plane perpendicular to the axis along which the gravity acts. Cases in which the actual unconstrained direction of motion requires a non zero component along the vector perpendicular to the ground plane are accounted for by the fixed-grasp estimator that becomes active after the start-up is completed. The force feedback update is depicted in Figure 4.1, where a bird's eye view of the drawer is given.

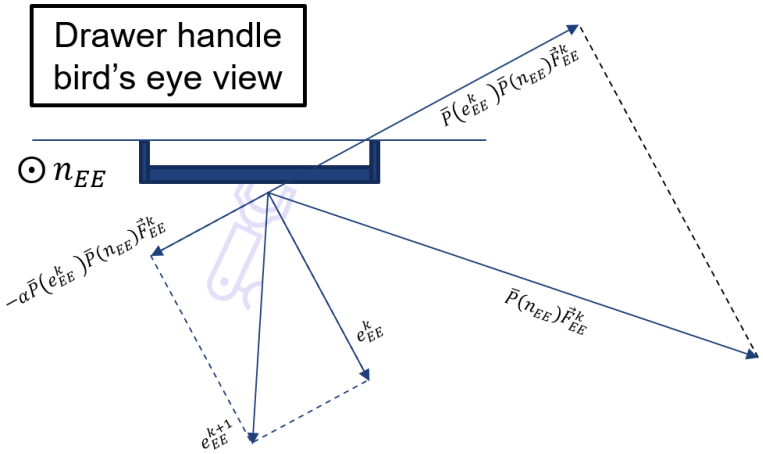


Figure 4.1: Force feedback based direction estimation.

Vector n_{EE} denotes the normal vector of the ground plane expressed in the end effector reference frame. The projection of the force exerted on the environment onto the ground plane is given by:

$$\bar{P}\left(n_{EE}\right)\vec{F}_{EE}^{k} \tag{4.1}$$

Furthermore, to get the projection on the direction perpendicular to the current unconstrained direction estimate we use:

$$\bar{P}\left(e_{EE}^{k}\right)\bar{P}\left(n_{EE}\right)\vec{F}_{EE}^{k} \tag{4.2}$$

This projection should be as small as possible. In order to decrease it after the current update step, we add a small component to our current estimate in the opposite direction of the current force projection. The updated value has the form:

$$\hat{e}_{EE}^{k+1} = e_{EE}^{k} - \alpha\frac{\left(\bar{P}\left(e_{EE}^{k}\right)\bar{P}\left(n_{EE}\right)\vec{F}_{EE}^{k}\right)}{\left\|\bar{P}\left(e_{EE}^{k}\right)\bar{P}\left(n_{EE}\right)\vec{F}_{EE}^{k}\right\|} \tag{4.3}$$

where α is a small fixed scaling coefficient. Note that after this kind of update we do not necessarily have a unit vector. To obtain the haptic based estimate we normalize the vector:

$$\hat{e}_{EE}^{k+1} = \frac{\hat{e}_{EE}^{k+1}}{\left\|\hat{e}_{EE}^{k+1}\right\|} \tag{4.4}$$

However, due to contact point switching problem, it is not enough to just rely on the force feedback during the update procedure. To make the procedure more stable, we also incorporate an estimator based on the fixed-grasp assumption as potential slipping causes smaller fluctuations in the measurements than the contact point switching.

4.2 Fixed-grasp Based Estimator

The fixed-grasp based estimator is activated only after the fixed length, cyclic buffer $\mathcal{L}_r$ has been sufficiently filled. At the end of each loop iteration during the start-up period, the end effector's 3D position $r_{I,EE}^k$ expressed in the world frame is stored in the buffer if:

$$\left\| r_{I,EE}^k - r_{I,EE}^{k-1} \right\| > \epsilon \tag{4.5}$$

where ϵ denotes a small fixed value that should prevent logging the successive positions if the end effector has not sufficiently moved between the two iterations. The end effector position is given by:

$$r_{I,EE} = r_{I,b} + C_{I,b} \cdot r_{b,EE} \tag{4.6}$$

where $r_{I,b}$ and $C_{I,b}$ denote the position and orientation matrix of the robot's base relative to the world frame and $r_{b,EE}$ is the end effector's relative position to the robot's base (expressed in the base frame). If the fixed-grasp assumption holds, then the buffer $\mathcal{L}_r$ actually contains the positions of the door handle. With this in mind, the problem of determining the unconstrained direction of motion can be reformulated into fitting an optimal line through these points and choosing the appropriate orientation of the line. The procedure is depicted in Figure 4.2. In case of a prismatic joint, the fitted line corresponds to the the actual trajectory of the door handle. For the revolute joint, this estimate corresponds to the current tangential direction to the circle along which the handle moves.

Vector of the optimal line fitted through a set of 3D points can be obtained using the Principal Component Analysis (PCA) [32]. The optimal line corresponds to the direction of the maximal variance of the data and will be determined by the first principal component. We start by rearranging the $\mathcal{L}_r$ buffer into a data matrix $X_{L_{\text{buff}}\times 3}$ and defining the data mean as:

$$C = [\bar{x}, \bar{y}, \bar{z}]^T = \frac{1}{L_{\text{buff}}} \sum_{i=0}^{L_{\text{buff}}-1} [\bar{x}_i, \bar{y}_i, \bar{z}_i]^T \tag{4.7}$$

Next, we need to center the data by subtracting the data mean from every row of the data matrix to obtain $\bar{X}_{L_{\text{buff}}\times 3}$. To find the first principal component we perform singular value decomposition (SVD) of the centered data matrix:

$$\bar{X}_{L_{\text{buff}}\times 3} = U \Sigma V^T \tag{4.8}$$

The vector of the optimal line now corresponds to the first column of the matrix V, denoted as $\vec{v}_1$. When using SVD to find the optimal line, we have to bear in

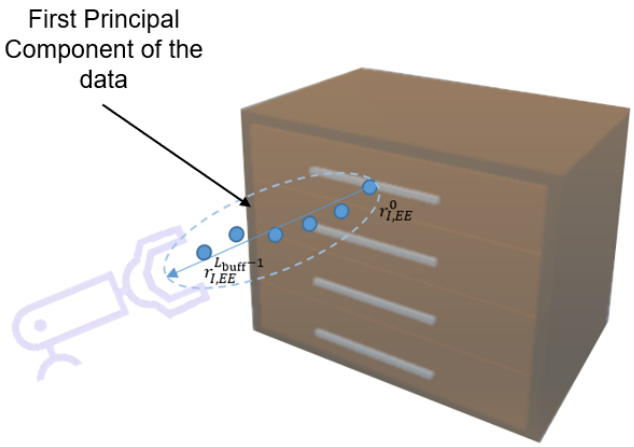


Figure 4.2: Fixed-grasp based direction estimation.

mind that the direction of the line is not uniquely determined. In fact, both $\vec{v}_1$ and $-\vec{v}_1$ are viable candidates and it is up to us to determine which direction of movement corresponds to our logged data. If we denote with $[x_0, y_0, z_0]^T$ the first point recorded in the buffer and with $[x_{L_{\text{buff}}-1}, y_{L_{\text{buff}}-1}, z_{L_{\text{buff}}-1}]^T$ the last one, then the fixed-grasp direction estimate is given by:

$$\hat{e}^{k+1}_{EE,PCA} = \vec{v}_1 \cdot \text{sgn}\left(\left\langle \vec{v}_1, \begin{bmatrix} x_{L_{\text{buff}}-1} \\ y_{L_{\text{buff}}-1} \\ z_{L_{\text{buff}}-1} \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \right\rangle\right) \tag{4.9}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar multiplication of two vectors.

Note that the fixed-grasp based estimator can also give a direction that is not in the plane parallel to the ground. This is necessary for the door opening trajectories that need to be executed in a plane perpendicular to the ground (such as dishwasher door). However, the performance of the fixed-based estimator is in every sense limited by the robot's capability to prevent relative motion between the gripper and the door handle. To derive a combined estimate we mix the outputs of the two estimators and filter it through the second order Butterworth filter to reduce the consequences of the contact point switching.

4.3 Filtering the Combined Measurement

To design a digital low-pass filtering sub-module, we have to provide the sampling period T_s and a desired digital cut off frequency f_c. The filtering is performed according to the Equation 4.10:

$$y[k] = a_{-1}y[k-1] + a_{-2}y[k-2] + bx[k] + b_{-1}x[k-1] + b_{-2}x[k-2] \tag{4.10}$$

where $y[k]$ denotes the discrete output of the filter, $x[k]$ denotes the discrete input to the filter (current combined estimate before filtering) and $y[k-2], y[k-1], x[k-2], x[k-1]$ denote outputs and inputs of the filter from the previous two discrete time steps. $a_{-1}, a_{-2}, b, b_{-1}$ and b_{-2} are the coefficients for the second order Butterworth filter [33] obtained by bilinear transformation [34] and pre-warping of the continuous

time transfer function of the filter given by:

$$G(s) = \frac{\omega_c^2}{s^2 + s\sqrt{2}\omega_c + \omega_c^2} \tag{4.11}$$

where ω_c is defined as:

$$\omega_c = \frac{2}{T_s} \tan \frac{2\pi f_c T_s}{2} \tag{4.12}$$

The transfer function of the digital Butterworth filter:

$$G(z) = \frac{Y(z)}{X(z)} \tag{4.13}$$

from which we can infer the required filter coefficients is obtained by substituting:

$$s = \frac{2}{T_s} \frac{1 - z^{-1}}{1 + z^{-1}} \tag{4.14}$$

into Equation 4.11. If we denote $\psi = \tan \frac{2\pi f_c T_s}{2}$, then the Butterworth coefficients are given by:

$$b = b_{-2} = \frac{\psi^2}{\psi^2 + \psi^2\sqrt{2} + 1} \tag{4.15}$$

$$b_{-1} = \frac{2\psi^2}{\psi^2 + \psi^2\sqrt{2} + 1} \tag{4.16}$$

$$a_{-1} = -\frac{2\psi^2 - 2}{\psi^2 + \psi^2\sqrt{2} + 1} \tag{4.17}$$

$$a_{-2} = -\frac{\psi^2 - \psi^2\sqrt{2} + 1}{\psi^2 + \psi^2\sqrt{2} + 1} \tag{4.18}$$

The low-pass filtering sub-module is designed so as to reduce the rapid changes in the combined unconstrained direction of motion estimate that would be imposed by the contact point switching and the force feedback estimator. To achieve so, we have to adequately tune the parameter f_c.

4.4 Putting it All Together

Now that we have described all the sub-modules of the online direction estimation procedure, we present the complete pipeline in the **Algorithm 2**.

At the beginning of each loop iteration we first append the current end effector position measurement to the end of the buffer. This is done only if the end effector has sufficiently moved relative to the last recorded position, as described by the Equation 4.5. Then, we calculate the haptic based estimate and normalize it to obtain a unit vector. If the current sampling iteration is higher than $N_{\text{start-up}}$, we also calculate the fixed-grasp estimate $\hat{e}_{EE,PCA}^{k+1}$ by performing the principal component analysis and combine it with the haptic based estimate by introducing a predefined, fixed, mixing coefficient m:

$$\hat{e}_{EE}^{k+1} \leftarrow m\hat{e}_{EE}^{k+1} + (1 - m)\hat{e}_{EE,PCA}^{k+1} \tag{4.19}$$

The purpose of the mixing coefficient m is to strike a balance between the sensitivity of the estimator to the external wrench acting on the end effector and the robustness provided by the fixed-grasp direction estimate. On the other hand, if the

Algorithm 2: Online direction of motion update step

input : k, $N_{\text{start-up}}$, e_{EE}^{k}, n_{EE}, F_{EE}^{k}, $\mathcal{L}_r$, $r_{I,EE}$, m
output: Updated unconstrained direction of motion e_{EE}^{k+1}

$\mathcal{L}_r$.append($r_{I,EE}$);
$\hat{e}_{EE}^{k+1} = e_{EE}^{k} - \alpha \frac{\left(\bar{P}\left(e_{EE}^{k}\right)\bar{P}(n_{EE})F_{EE}^{k}\right)}{\left\|\bar{P}\left(e_{EE}^{k}\right)\bar{P}(n_{EE})F_{EE}^{k}\right\|}$;
$\hat{e}_{EE}^{k+1} = \frac{\hat{e}_{EE}^{k+1}}{\left\|\hat{e}_{EE}^{k+1}\right\|}$;

if $k > N_{start\text{-}up}$ **then**
  $\hat{e}_{EE,PCA}^{k+1} = \text{PCA}(\mathcal{L}_r)$;
  $\hat{e}_{EE}^{k+1} \leftarrow m\hat{e}_{EE}^{k+1} + (1-m)\hat{e}_{EE,PCA}^{k+1}$;
  $\hat{e}_{EE}^{k+1} = \frac{\hat{e}_{EE}^{k+1}}{\left\|\hat{e}_{EE}^{k+1}\right\|}$;

$e_{EE}^{k+1} = \text{Filter}(\hat{e}_{EE}^{k+1}, \hat{e}_{EE}^{k}, \hat{e}_{EE}^{k-1}, e_{EE}^{k}, e_{EE}^{k-1})$;
$e_{EE}^{k-1} \leftarrow e_{EE}^{k}$;
$e_{EE}^{k} \leftarrow e_{EE}^{k+1}$;
$\hat{e}_{EE}^{k-1} \leftarrow \hat{e}_{EE}^{k}$;
$\hat{e}_{EE}^{k} \leftarrow \hat{e}_{EE}^{k+1}$;

current sampling iteration is not higher than $N_{\text{start-up}}$, the mixing step is skipped. In either case, the normalized current estimate is fed as an input to the filtering sub-module and the previous values of the filter's inputs and corresponding outputs are updated. Finally, the output of the filter is transferred to the end effector velocity planner module as our current estimate of the unconstrained direction of motion.

Having defined the direction in which the end effector should move, in the next chapter we proceed to thoroughly describe all the sub-modules of the end effector velocity planner module.

Chapter 5

End Effector Velocity Planner

The updated estimate received from the online direction estimation module determines the total linear and angular velocity vectors for the current loop iteration. The output of the velocity planner module is given by:

$$\begin{bmatrix} \dot{q}^* \\ v_b^{base} \end{bmatrix} \tag{5.1}$$

and has to satisfy the constraints imposed by the robot. The complete schematic representation of the module is presented in the Figure 5.1.

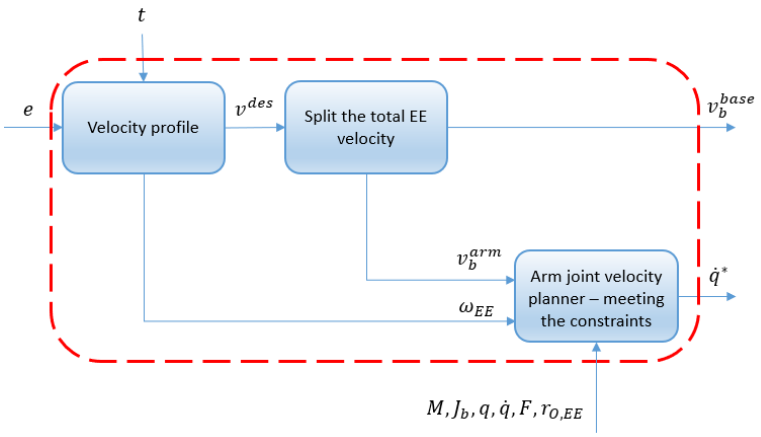


Figure 5.1: Block diagram of the velocity planner module.

There are three sub-modules encapsulated within the velocity planner module:

- **Velocity profile:** This sub-module is in charge of ensuring that the magnitude of the total end effector linear velocity is smaller during the start-up period of the algorithm since we assume that during that time the direction estimate has not yet converged. Furthermore, it should also ensure the velocity profile is smooth and differentiable at all times. If we are operating a rotational type of joint mechanism, then this sub-module also calculates the desired angular velocity such that we align the appropriate end effector axis with the current unconstrained direction of motion estimate. Since the estimate is changing over time, this module should provide slow updates of the

end effector's orientation in order to avoid oscillations while still being able to help the robot perform the door opening procedure.

- **Total velocity splitting module:** This sub-module utilizes an optimization procedure to split the total velocity into a component to be commanded to the robot base v_b^{base} and a component that determines the relative motion between the end effector and the base v_b^{arm}. The main idea behind this sub-module is to increase the manipulability of the robotic arm by utilizing the additional DOFs available through the mobile base. This is done in order to increase the workspace of the robot and to alleviate the loss of redundancy imposed by fixing the grasp. Since one of our goals is to achieve fast real-time operation, we will investigate several problem formulations that match the commercially available optimization software in order to identify the one that gives the best trade-off between repeatability and the execution speed.

- **Hardware constraint aware planning module:** To make sure we send optimal joint velocity commands within hardware constraints, we perform convex linearly constrained quadratic programming. The constraints that we take into account are the joint position limits, joint velocity limits, joint acceleration limits and the torque limits.

5.1 Linear Velocity Time Profile

Given the total number of loop iterations to be performed, we design a time profile of the linear velocity magnitude so as to resemble the one presented in Figure 5.2.

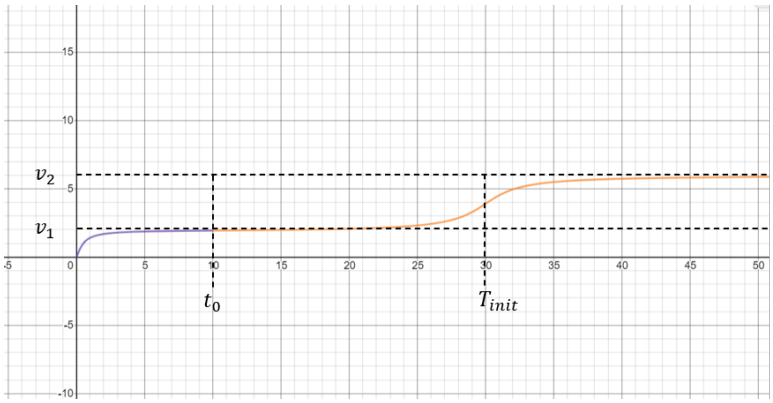


Figure 5.2: Time profile of the linear velocity magnitude.

To make the profile smooth and differentiable we combine two arctan segments into a function whose inputs are given by:

$$v_1, v_2, t_0, T_{\text{init}}\ , \alpha_{\text{init}} \text{ and } \alpha_{\text{regular}}$$

Here, v_1 denotes the velocity magnitude during the start-up period, v_2 is the velocity magnitude during the remaining part of the procedure, $T_{\text{init}} = N_{\text{start-up}}$, t_0 is the time point when the profile switches from one arctan segment to the other (in Figure 5.2, this corresponds to switching from the purple to the orange segment of the function) and α_{init} , α_{regular} are the parameters that define the shape of the function such that it is given by:

$$|v|(t) = \begin{cases} v_1 \frac{2}{\pi} \arctan\left(\alpha_{\text{init}}\ t\right) & \text{for } t < t_0 \\ a_1 \frac{2}{\pi} \arctan\left(\alpha_{\text{regular}}\ \left(t - T_{\text{init}}\ \right)\right) + a_2 & \text{for } t \geq t_0 \end{cases} \tag{5.2}$$

Values of the coefficients a_1 and a_2 are determined such that the following two requirements are satisfied:

1. $\lim_{t \to t_0^-} |v|(t) = \lim_{t \to t_0^+} |v|(t)$
2. $\lim_{t \to \infty} |v|(t) = v_2$

Combining the two equations yields the following solutions for a_1 and a_2:

$$a_1 = \frac{v_2 - v_1 \frac{2}{\pi} \arctan\left(\alpha_{\text{init}}\ t_0\right)}{1 - \frac{2}{\pi} \arctan\left(\alpha_{\text{regular}}\ \left(t_0 - T_{\text{init}}\ \right)\right)} \tag{5.3}$$

$$a_2 = v_2 - a_1 \tag{5.4}$$

By construction, this kind of velocity time profile provides smooth transitions between the initial start-up period with smaller velocity magnitude to the second regime in which the velocity magnitude is higher.

5.2 Angular Velocity Planner

When dealing with rotational joint mechanisms, it is important to also adjust the orientation of the end effector so that the door opening procedure can be more easily completed. In more severe cases, not adjusting the orientation could lead to slipping of the end effector or collision with the door. If we adopt the end effector frame axis notation introduced in [30], then the goal of this sub-module is to iteratively achieve alignment between end effector's z axis and the door plane normal vector. In our setup, the current estimate of the door plane normal vector corresponds to the current estimate of the unconstrained direction of motion. The angular velocity planning step is depicted in Figure 5.3.

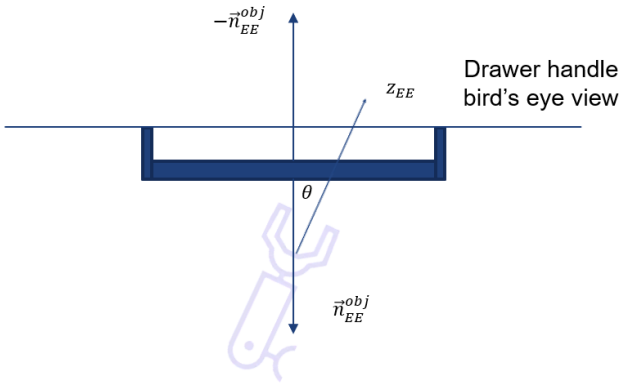


Figure 5.3: Angular velocity planning sub-module.

Here, θ represents the angular displacement from the desired pose θ_{des} that should be compensated. It can be calculated using:

$$\theta = \arccos\left(\left\langle -n_{EE}^{obj}, z_{EE} \right\rangle\right) - \theta_{des} \tag{5.5}$$

With this in mind, we plan the angular velocity as:

$$\omega_{EE} = k \left(\arccos \left(\left\langle -n_{EE}^{obj}, z_{EE} \right\rangle \right) - \theta_{des} \right) \left(z_{EE} \times \left(-n_{EE}^{obj} \right) \right) \tag{5.6}$$

where k is a tuning coefficient, n_{EE}^{obj} is the current unconstrained direction of motion estimate and $\theta_{des} = 0$. Increasing the value of coefficient k leads to faster compensation of the angular displacement. However, fixing it to a large value may lead to undesired behaviour that could further lead to oscillations and instability. Since we do not have the actual value of the door plane normal vector but only its estimate, we set k to be a smaller value to prevent undesired behaviour. Furthermore, we choose not to command the angular velocity during the start-up period in which the unconstrained direction of motion estimate has not yet converged.

5.3 Splitting the Total Desired End Effector Velocity

Let us denote with

$$v_{des} = |v|(t)e_k = \begin{bmatrix} v_{des}^x \\ v_{des}^y \\ v_{des}^z \end{bmatrix}$$

the total desired end effector velocity based on the model fitting procedure (expressed in the base frame). The purpose of this sub-module would be to separate v_{des} into two components:

$$v_{des} = v_b^{base} + v_b^{arm} \tag{5.7}$$

such that v_b^{base} denotes the desired linear velocity component to be commanded to the mobile base and v_b^{arm} denotes the relative velocity between the arm and the mobile base. Note that the base can only contribute in the x and y directions of the base frame so the movement in the z direction must completely be determined by $v_b^{z,Arm}$. Thus, we introduce two variables Δ_1 and Δ_2 that will determine the split of the total desired end effector velocity in the x and y directions. The split is done via optimization procedure such that the following two aspects are taken into account:

1. If the distance in the x-y plane between the end effector frame origin and the base frame origin decreases, then the relative linear velocity between the arm and the base, v_b^{arm}, should decrease as well. This is done to prevent the end effector from approaching too close to the base and hence, effectively reduces the risk of self collision.

2. We try to plan for the joint positions that are as far away as possible from the joint limits so that we increase the manipulability of the robot.

If we denote the distance in the x-y plane between the end effector frame origin and the base frame origin with R, then we have:

$$R = \sqrt{x_{b,EE}^2 + y_{b,EE}^2} \tag{5.8}$$

where $T_{b,EE}$ represents the homogeneous transformation from the end effector frame to the base frame and contains the information about $x_{b,EE}$ and $y_{b,EE}$.

The first aspect can be addressed by introducing a fixed limit distance R_{lim} and by defining an adaptive scaling coefficient as:

$$\lambda = \text{sgn}\left(R - R_{\text{lim}}\right) \left(1 - e^{-\alpha |R - R_{\text{lim}}|} \right) \tag{5.9}$$

and the arm velocity component as:

$$v_b^{arm} = \begin{bmatrix} \lambda\Delta_1 \\ \lambda\Delta_2 \\ v_{des}^z \end{bmatrix} \tag{5.10}$$

Combining equations 5.7 and 5.10 yields the corresponding base velocity:

$$v_b^{base} = \begin{bmatrix} v_{des}^x - \lambda\Delta_1 \\ v_{des}^y - \lambda\Delta_2 \\ 0 \end{bmatrix} \tag{5.11}$$

Note that the adaptive scaling coefficient satisfies $\lambda \in [-1, 1]$ and $\lim_{R \to R_{\text{lim}}^-} \lambda = \lim_{R \to R_{\text{lim}}^+} \lambda = 0$. Furthermore, when $R < R_{\text{lim}}$, the scaling coefficient encourages the relative velocity between the arm and the base to be in the opposite direction compared to the case when $R > R_{\text{lim}}$.

The second aspect is addressed by introducing $\dot{\bar{q}}$. It corresponds to the joint velocity vector that would drive the joints to be closer to the middle point of their position range denoted as $\bar{q}$. If the current joint position is given by q_k, then the velocity vector that would drive the joint positions towards the middle points of the respective ranges in one sampling interval is given by:

$$\dot{\bar{q}} = \frac{\bar{q} - q_k}{\Delta T} \tag{5.12}$$

Since we are doing one step ahead planning, we introduce a scaling coefficient γ so that we only take a small step towards the middle joint position point $\bar{q}$. If we also account for the joint velocity limits, we finally define $\dot{\bar{q}}$ as:

$$\dot{\bar{q}} = \begin{cases} \min\left\{\dot{q}_{\max}, \gamma \frac{\bar{q} - q_k}{\Delta T}\right\}, & \text{if } \bar{q} - q_k > 0 \\ \\ \max\left\{\dot{q}_{\min}, \gamma \frac{q_k - \bar{q}}{\Delta T}\right\}, & \text{if } \bar{q} - q_k < 0 \end{cases} \tag{5.13}$$

where the minimum, absolute value and the comparisons are all done element wise.

We design the objective function to be minimized such that we try to comply with the requirements to:

- enforce the relative velocity vector v_b^{arm} to be as close as possible to the Cartesian velocity vector determined by $\dot{\bar{q}}$ and given by $J_{b,EE}^{lin} \cdot \dot{\bar{q}}$
- minimize the velocity component to be commanded to the mobile base

while simultaneously keeping the Euclidean norm of the vector $\Delta = [\Delta_1, \Delta_2]^T$ constrained. There are several commercially available optimization solvers that can be used. They primarily differ in the type of problems they can solve and, consequently, the average execution time. In this thesis, we will investigate three plausible problem formulations such that each of them is particularly suitable for a certain solver. At the end, we will choose the one that gives the best trade-off between the average execution time, the number of successfully completed runs and the overall average performance during a run.

5.3.1 Quadratically Constrained Convex Optimization

We first investigate the problem formulation suitable for one of the most common optimization libraries in Python, **scipy.optimize**. This library imposes no restrictions on the optimization objective and constraints and as such is capable of solving

both convex and non-convex optimization problems with huge variety of constraints. This, however, comes at the expense of higher execution times compared to the solvers optimized for a particular set of problems. We define the objective function as:

$$\min_{\Delta_1,\Delta_2} c_1 \left\| \begin{bmatrix} v_{des}^x - \lambda\Delta_1 \\ v_{des}^y - \lambda\Delta_2 \\ 0 \end{bmatrix} \right\|_2 + c_2 \left\| \begin{bmatrix} \lambda\Delta_1 \\ \lambda\Delta_2 \\ v_{des}^z \end{bmatrix} - J_{b,EE}^{lin}\dot{\bar{q}} \right\|_2 \tag{5.14}$$

and the constraint as:

$$\Delta_1^2 + \Delta_2^2 \leq \beta^2 |v|^2(k) \tag{5.15}$$

Note that the objective function is convex, the constraint is quadratic and c_1, c_2 and β are fixed, a priori defined constants. The first term in the objective function tries to minimize the velocity of the mobile base while the second term enforces the relative velocity vector to be as close as possible to the Cartesian velocity determined by $\dot{\bar{q}}$. Equation 5.14 can be reformulated in a more compact form:

$$\min_{\Delta} c_1 \left\| \hat{v}_{des} - \lambda\mathbb{I}\Delta \right\|_2 + c_2 \left\| \Delta - \hat{v}_{mid} \right\|_2 \tag{5.16}$$

where $\hat{v}_{des} = [v_{des}^x, v_{des}^y]^T$ and $\lambda\hat{v}_{mid}$ is equal to a vector containing the first two elements of the vector $J_{b,EE}^{lin}\dot{\bar{q}}$. We denote this problem formulation as quadratically constrained convex optimization (QCCO).

Equations 5.15 and 5.16 are now fed to the optimizer that eventually outputs the optimal value Δ^*. When designing the objective function, we relied only on the Euclidean norm of the vectors. This can be utilized to reformulate the problem into a second order cone program described in the following subsection.

5.3.2 Second Order Cone Program

There exists a specialized **cvxopt** library [35] suitable for solving the second order cone programming (SOCP) problems. Standard problem formulation is defined as:

$$\begin{aligned} \min_{x} \quad & f^T x \\ \text{s.t.} \quad & \left\| A_k x + b_k \right\|_2 \leq c_k^T x + f_k, \\ & Fx = g \end{aligned}$$

Taking into consideration the equations 5.15 and 5.16, and by introducing additional optimization variables x_1 and x_2, we can reformulate the original problem from the Section 5.3.1 into a second order cone program.

The objective function can be given by:

$$\min_{\tilde{\Delta}=[x_1,x_2,\Delta]} \begin{bmatrix} c_1 & c_2 & 0 & 0 \end{bmatrix} \tilde{\Delta} \tag{5.17}$$

with f being:

$$f = \begin{bmatrix} c_1 & c_2 & 0 & 0 \end{bmatrix}^T \tag{5.18}$$

while the three constraints are given as:

$$\left\| \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & \lambda\mathbb{I}_{2\times2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} - \hat{v}_{des} \right\|_2 \leq \begin{bmatrix} 1 & 0 & 0_{1\times2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} \tag{5.19}$$

$$\left\| \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & \mathbb{I}_{2\times2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} - \hat{v}_{mid} \right\|_2 \leq \begin{bmatrix} 0 & 1 & 0_{1\times2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} \tag{5.20}$$

$$\left\| \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & \mathbb{I}_{2\times2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} \right\|_2 \leq \beta|v|(k) \tag{5.21}$$

The corresponding matrices that determine the constraints of the second order cone program formulation are given by:

$$A_1 = \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & \lambda\mathbb{I}_{2\times2} \end{bmatrix}; \quad b_1 = -\hat{v}_{des}; \quad c_1 = \begin{bmatrix} 1 \\ 0 \\ 0_{2\times1} \end{bmatrix}; \quad d_1 = 0 \tag{5.22}$$

$$A_2 = \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & \mathbb{I}_{2\times2} \end{bmatrix}; \quad b_2 = -\hat{v}_{mid}; \quad c_2 = \begin{bmatrix} 0 \\ 1 \\ 0_{2\times1} \end{bmatrix}; \quad d_2 = 0 \tag{5.23}$$

$$A_3 = \begin{bmatrix} 0_{2\times1} & 0_{2\times1} & \mathbb{I}_{2\times2} \end{bmatrix}; \quad b_3 = 0_{2\times1}; \quad c_3 = 0_{4\times1}; \quad d_3 = \beta|v|(k) \tag{5.24}$$

However, to use the **cvxopt** solver we need to write the problem in a different form. According to [35], the problem should take the form:

$$\begin{aligned} \min_{x,\,s} \quad & f^T x \\ \text{s.t.} \quad & G_k x + s_k = h_k, \\ & s_{k0} \geq \|s_{k1}\|_2, \\ & Fx = g, \\ & k = 1, \ldots, M \end{aligned}$$

where the optimization variables s_k are defined by: $s_k = \begin{bmatrix} s_{k0} \\ s_{k1} \end{bmatrix}$. In our case $M = 3$ so for every $k \in \{1, 2, 3\}$ we have:

$$s_k = \begin{bmatrix} s_{k0} \\ s_{k1} \end{bmatrix} = \begin{bmatrix} c_k^T x + d_k \\ A_k^T x + b_k \end{bmatrix} = \begin{bmatrix} c_k^T \\ A_k \end{bmatrix} x + \begin{bmatrix} d_k \\ b_k \end{bmatrix} = -G_k x + h_k \tag{5.25}$$

which yields:

$$G_k = -\begin{bmatrix} c_k^T \\ A_k \end{bmatrix} \text{ and } h_k = \begin{bmatrix} d_k \\ b_k \end{bmatrix} \tag{5.26}$$

Finally, combining equations 5.22, 5.23, 5.24 and 5.26 yields the complete formulation of the problem.

The first constraint is now defined via:

$$G_1 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix} \quad \text{and} \quad h_1 = \begin{bmatrix} 0 \\ -\hat{v}_{des}^x \\ -\hat{v}_{des}^y \end{bmatrix} \tag{5.27}$$

The second constraint is defined via:

$$G_2 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad \text{and} \quad h_2 = \begin{bmatrix} 0 \\ -\hat{v}_{mid}^{x} \\ -\hat{v}_{mid}^{y} \end{bmatrix} \tag{5.28}$$

The third constraint is defined via:

$$G_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad \text{and} \quad h_3 = \begin{bmatrix} \beta \left| v \right| (k) \\ 0 \\ 0 \end{bmatrix} \tag{5.29}$$

Equations 5.18, 5.27, 5.28 and 5.29 are fed to the optimizer that outputs the optimal value $\widetilde{\Delta}^*$. Given that the **cvxopt** library is specialized for solving the second order cone programming, we expect to experience shorter average execution time per iteration. Finally, we can also simplify the original optimization problem into a linearly constrained quadratic program (LCQP) and use a specialized library deemed faster than the two previously introduced.

5.3.3 Linearly Constrained Quadratic Program

The simplest problem formulation consists of a quadratic objective with linear constraints. To solve such a problem, one could utilize the fast optimization library **quadprog** [36] suitable only for strictly convex optimization problems. The problem description required by the optimizer is given by:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T G x - a^T x \\ \text{s.t.} \quad & C^T x \geq b \end{aligned}$$

Unfortunately, though the objective function given by Equation 5.16 is convex, it is not quadratic. Furthermore, the constraint given by 5.15 is quadratic. To match the required problem description, we reformulate the objective as:

$$\min_{\Delta} c_1 \left\| \hat{v}_{des} - \lambda \mathbb{I} \Delta \right\|_2^2 + c_2 \left\| \Delta - \hat{v}_{mid} \right\|_2^2 \tag{5.30}$$

By utilizing the fact that $\|a\|_2^2 = a^T a$ and regrouping the terms, we can write the objective as:

$$\min_{\Delta} \Delta^T \left[\left(c_2 + c_1 \lambda^2 \right) \mathbb{I} \right] \Delta - 2 \left(c_1 \lambda \hat{v}_{des}^T + c_2 \hat{v}_{mid}^T \right) \Delta + c_1 \hat{v}_{des}^T \hat{v}_{des} + c_2 \hat{v}_{mid}^T \hat{v}_{mid} \tag{5.31}$$

This is now equivalent to:

$$\min_{\Delta} \frac{1}{2} \Delta^T \left[\left(c_2 + c_1 \lambda^2 \right) \mathbb{I} \right] \Delta - \left(c_1 \lambda \hat{v}_{des} + c_2 \hat{v}_{mid} \right)^T \Delta \tag{5.32}$$

which finally yields the matrix G:

$$G = \left(c_2 + c_1 \lambda^2 \right) \mathbb{I} \tag{5.33}$$

and the vector a:

$$a = c_1 \lambda \hat{v}_{des} + c_2 \hat{v}_{mid} \tag{5.34}$$

Since **quadprog** only solves strictly convex problems, we have to make sure that in every iteration we have:

$$G \succ 0$$

Since $c_1 > 0$ and $c_2 > 0$, this is obviously true for every λ.

In order to address the problem of linear constraints, we have to approximate $\|\Delta\|_2 \leq \beta|v|(k)$ with a set of linear inequalities. Since the bound on the 2-norm imposes a circle as a feasible set for the optimization variable Δ, we approximate the area of a circle with the radius $\beta|v|(k)$ with the area of a regular N_p-sided polygon circumscribed around it. This is done by defining a set of linear inequalities:

$$n_i^T \Delta \leq d_i \tag{5.35}$$

where $i \in \{0, 1, \ldots, N_\mathrm{p} - 1\}$. Each pair (n_i, d_i) is now determined by the corresponding side of the polygon. Figure 5.4 depicts the case when $N_p = 8$.

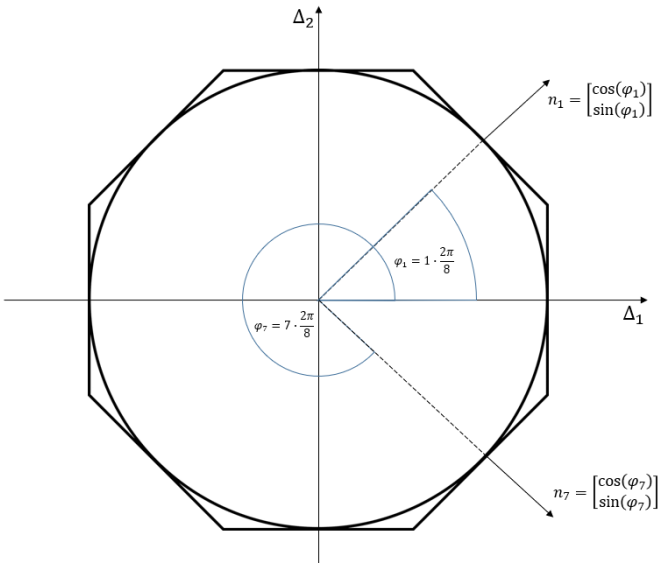


Figure 5.4: Approximating $\|\Delta\|_2$ with a set of linear constraints for $N_p = 8$.

Vectors n_i are given by:

$$n_i = \begin{bmatrix} \cos(\varphi_i) \\ \sin(\varphi_i) \end{bmatrix} \tag{5.36}$$

where the angles φ_i are given by:

$$\varphi_i = i \frac{2\pi}{N_p} \tag{5.37}$$

and scalars d_i by:

$$d_i = \beta|v|(k) \tag{5.38}$$

To match the description required by the **quadprog** library we rearrange and stack the individual constraints given by 5.35 so as to form:

$$C = \begin{bmatrix} -n_0^T \\ -n_1^T \\ \vdots \\ -n_{N_p-1}^T \end{bmatrix}^T \quad \text{and} \quad b = \begin{bmatrix} -d_0 \\ -d_1 \\ \vdots \\ -d_{N_p-1} \end{bmatrix} \tag{5.39}$$

In this section, we presented three different problem formulations that should all provide a split of the total desired linear end effector velocity. Regardless of the method that is being used, we enforce encapsulation of the code such that the "plug-and-play" feature is provided. This is achieved by ensuring that the inputs and the outputs of each end effector velocity splitting module are always the same. In order to send an actual joint velocity command to the robot arm, the output v_b^{arm} of the velocity splitting module is transferred as the input to the module that calculates desired joint velocities in compliance with the hardware constraints of the robot. This sub-module will be thoroughly described in the following section.

5.4 Joint Velocities Within Hardware Constraints

There are four types of hardware constraints that we have to take into account:

1. Joint positions at the end of the loop iteration, estimated via forward integration, have to satisfy $q_{\min} \leq q \leq q_{\max}$, where $q_{\min}$ and $q_{\max}$ represent the vectors of the lower and the upper joint position boundaries.

2. Commanded joint velocities have to satisfy $\dot{q}_{\min} \leq \dot{q} \leq \dot{q}_{\max}$, where $\dot{q}_{\min}$ and $\dot{q}_{\max}$ represent the vectors of the lower and the upper joint velocity boundaries.

3. Joint accelerations estimated via finite difference scheme have to satisfy $\ddot{q}_{\min} \leq \ddot{q} \leq \ddot{q}_{\max}$, where $\ddot{q}_{\min}$ and $\ddot{q}_{\max}$ represent the vectors of the lower and the upper joint acceleration boundaries.

4. Torques calculated based on the joint impedance control have to satisfy $\tau_{\min} \leq \tau \leq \tau_{\max}$, where $\tau_{\min}$ and $\tau_{\max}$ represent the vectors of the lower and the upper joint torque boundaries.

All above described constraints fall under the category of linear constraints so in order to optimally plan within them, we formulate a convex linearly constrained quadratic program. Just like in the Section 5.3.3, we will utilize the **quadprog** library for its time efficiency. Since we have two inequalities for each of the four different constraint types, there are eight constraints in total that we have to incorporate. Let us first denote with:

$$\zeta = [\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_7] \tag{5.40}$$

the optimization variable. The objective function can now be formulated as:

$$\min_{\zeta} \left\| J_{b,EE}\zeta - \begin{bmatrix} v_b^{Arm} \\ \omega_{EE} \end{bmatrix} \right\|_2^2 = \|J_{b,EE}\zeta - v_{\text{total}}^{arm}\|_2^2 \tag{5.41}$$

Similarly to what has been done in the Section 5.3.3, we can transform the objective into:

$$\min_{\zeta} \zeta^T J_{b,EE}^T J_{b,EE}\zeta - 2(v_{\text{total}}^{arm})^T J_{b,EE}\zeta + (v_{\text{total}}^{arm})^T v_{\text{total}}^{arm} \tag{5.42}$$

Once again, we can remove the constant term from the objective function and scale it to match the required problem formulation:

$$\min_{\zeta} \frac{1}{2}\zeta^T J_{b,EE}^T J_{b,EE}\zeta - (v_{\text{total}}^{arm})^T J_{b,EE}\zeta \tag{5.43}$$

From the Equation 5.43 we can now extract the corresponding G matrix:

$$G = J_{b,EE}^T J_{b,EE} \tag{5.44}$$

and the a vector:

$$a = J_{b,EE}^T v_{\text{total}}^{arm} \tag{5.45}$$

Since the matrix G is symmetric, the condition $G \succ 0$ imposed by the **quadprog** optimizer is satisfied. If q_k denotes the current joint positions, then the joint position constraints can be expressed as:

$$q_{\min} \leq q_k + \zeta \Delta T \leq q_{\max} \tag{5.46}$$

Joint velocity constraints are simply given by:

$$\dot{q}_{\min} \leq \zeta \leq \dot{q}_{\max} \tag{5.47}$$

Similarly, if $\dot{q}_k$ denotes the current joint velocities, then the joint acceleration constraints are given by:

$$\ddot{q}_{\min} \leq \frac{\zeta - \dot{q}_k}{\Delta T} \leq \ddot{q}_{\max} \tag{5.48}$$

Equations 5.46, 5.47 and 5.48 can be combined into two joint constraints given by:

$$\mathbb{I}_{7\times 7}\zeta \geq \max\left\{\frac{q_{\min} - q_k}{\Delta T},\ \dot{q}_{\min},\ \dot{q}_k + \ddot{q}_{\min}\Delta T\right\} \tag{5.49}$$

$$-\mathbb{I}_{7\times 7}\zeta \geq -\min\left\{\frac{q_{\max} - q_k}{\Delta T},\ \dot{q}_{\max},\ \dot{q}_k + \ddot{q}_{\max}\Delta T\right\} \tag{5.50}$$

where maximum and minimum are done element wise. Given that we use the internal joint impedance control, the torque constraint can be written as:

$$\tau_{\min} \leq K_p\left(q^{des} - q_k\right) + K_d\left(\zeta - \dot{q}_k\right) + b + g \leq \tau_{\max} \tag{5.51}$$

where b denotes the Coriolis component, g the gravity component and matrices K_p and K_d are given as:

$$\begin{aligned} K_p &= \text{diag}\left(600, 600, 600, 600, 250, 150, 50\right) \\ K_d &= \text{diag}\left(50, 50, 50, 20, 20, 20, 10\right) \end{aligned} \tag{5.52}$$

These are the default values of the K_p and K_d matrices provided in the joint impedance control example of the **franka_control** package.

If we set $q^{des} = q_k + \zeta\Delta T$, then the torque constraint can be written as:

$$\tau_{\min} + K_d\dot{q}_k - b - g \leq \left[K_p\Delta T + K_d\right]\zeta \leq \tau_{\max} + K_d\dot{q}_k - b - g \tag{5.53}$$

Equation 5.53 yields two additional constraints in the form required by the **quadprog** library:

$$\left[K_p\Delta T + K_d\right]\zeta \geq \tau_{\min} + K_d\dot{q}_k - b - g \tag{5.54}$$

$$-\left[K_p \Delta T + K_d\right] \zeta \geq -\tau_{\max} - K_d \dot{q}_k + b + g \tag{5.55}$$

Finally, by stacking individual constraints given by equations 5.49, 5.50, 5.54 and 5.55 we obtain the matrix C as:

$$C = \begin{bmatrix} \mathbb{I}_{7\times 7} \\ -\mathbb{I}_{7\times 7} \\ K_p \Delta T + K_d \\ -K_p \Delta T - K_d \end{bmatrix}^T \tag{5.56}$$

and the vector b as:

$$b = \begin{bmatrix} \max\left\{\frac{q_{\min} - q_k}{\Delta T}, \dot{q}_{\min}, \dot{q}_k + \ddot{q}_{\min}\Delta T\right\} \\ -\min\left\{\frac{q_{\max} - q_k}{\Delta T}, \dot{q}_{\max}, \dot{q}_k + \ddot{q}_{\max}\Delta T\right\} \\ \tau_{\min} + K_d \dot{q}_k - b - g \\ -\tau_{\max} - K_d \dot{q}_k + b + g \end{bmatrix} \tag{5.57}$$

Finally, matrices G, a, C and b are given as the inputs to the optimizer that, as a result, outputs the optimal joint velocities ζ^* calculated within the constraints imposed by the robotic arm. In this thesis we do not address any particular constraints regarding the velocities commanded to the mobile base as we plan to send small velocity commands that are most certainly in compliance with the limits imposed by the ridgeback mobile base.

Before we proceed to analyse the results obtained in the simulated and real environments, a thorough description of the communication between the individual components that will be deployed on the real robot is needed. We present this in the following chapter.

Chapter 6

System Integration

As we have previously mentioned, the complete method is modular, allowing us to design, exchange and test individual components within our framework. In PyBullet simulation, all the information regarding the current state of the robot and the operated mechanisms is provided by the built-in functions of the simulator (based on the URDF models of the objects included in the simulation). Furthermore, the execution here is synchronized, meaning that we perform one step of the simulation once all the computation is completed, which does not correspond to the real world scenario. In Gazebo and on the real robot, this is not the case. Different computations and the interaction with the environment happen asynchronously and the state of the robotic arm, the mobile base and the gripper are received separately through ROS topics and ROS services. Since we have to manage communication between three individual devices, the user interface of the whole body control method is organized in a form of a state machine. On the real robot, different states are in charge of sending the non-realtime commands, reading the current states of the system components and updating the desired reference values. In the Gazebo simulation, we have only one device (user's computer) running all the code. Therefore, in the following two sections we present different implementation aspects that are taken into account when designing the communication system for the Gazebo simulation and for the real robot.

6.1 Communication in the Gazebo Simulation

The complete system in the Gazebo simulation is presented in Figure 6.1. The state machine node, Gazebo simulation and the ROS Controller nodes all run on the same device. We design the ROS Gripper Service node, the Franka State Service node and the Franka ROS Controller node whereas the Ridgeback Controller and the Ridgeback State nodes have already been provided to us.

In total, four states are visited during one run of the complete procedure:

- **Starting state:** It creates the online direction estimation object, the controller optimizer object and the closed loop planner object. Furthermore, it connects the planner with the PyBullet library in order to allow the state machine to receive the information about the end effector jacobian, mass matrix, the Coriolis and the gravity components. Afterwards, this state virtually sets the end effector and the stiffness frames (which is required on the real robot). Finally, it waits until the user is ready to close the gripper and switches to the next state.

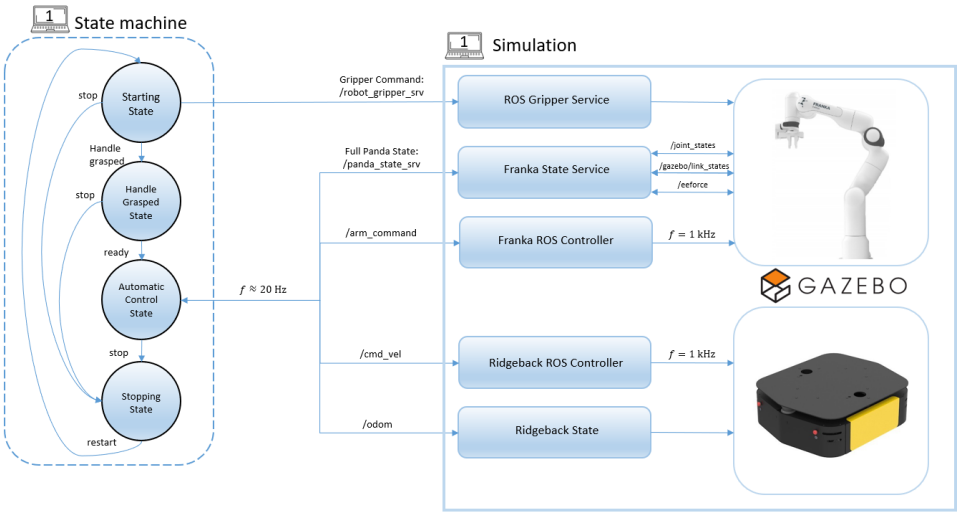


Figure 6.1: Communication in the Gazebo simulation.

- **Handle Grasped State:** In this state, user is required to send a command for closing the gripper. Before the transition to the next state is accomplished, this state waits for the user's input to initiate the automatic control regime.
- **Automatic Control State:** This state performs N_{total} control loop iterations if not interrupted by the user. It communicates with an approximate frequency of $f = 20\,\text{Hz}$ with the Franka State Service node, Franka Ros Controller and the Ridgeback Controller. At each time step, it requests a full state of the robotic arm, collects the information from the URDF model of the robot using PyBullet, performs the two optimizations and publishes the base and the arm commands.
- **Stopping state:** A transition to this state happens either upon the completion of the automatic door opening procedure or when the user gives a command to abort the current run of the program while being in any of the previously described states. In either case, this state publishes zero velocity commands to the arm and the mobile base.

A complete list of ROS topics and services used in the Gazebo simulation is given in Table 6.1.

Table 6.1: ROS topics and services used in the Gazebo simulation.

ROS topics and services	Description
/panda_state_srv	receives the full state of the robot arm
/robot_gripper_srv	closes the gripper and sets its frames
/joint_states	receives q, $\dot{q}$ and τ
/gazebo/link_states	receives position and velocity of all links
/eeforce	receives simulated F_{EE}
/odom	receives $T_{O,base}$ and v_b^{base}
/arm_command	publishes desired joint velocities
/cmd_vel	publishes desired mobile base velocity

We keep the same structure of the state machine and the two ROS services when

writing the code to be deployed on the real hardware. What differs is the set of ROS topics used and the way the state machine receives the information about the state of the robotic arm. This is described in the following section.

6.2 Communication on the Real Robot

The complete system deployed on the actual hardware is presented in Figure 6.2.

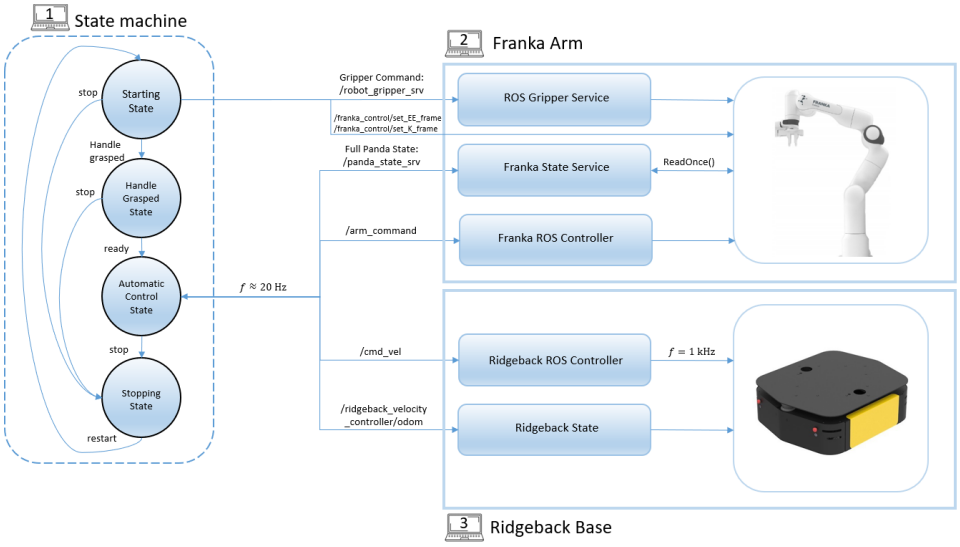


Figure 6.2: Communication in the Gazebo simulation.

Compared to the system used in the Gazebo simulation, the main difference stems from the fact that we now have three separate computers running several ROS nodes. The state machine is once again located on the user's computer, but the ROS Gripper Service, Franka State Service and Franka ROS Controller are deployed on the Franka arm computer. Similarly, the Ridgeback ROS Controller and the Ridgeback State node have to be located on the Ridgeback computer. In order for these three computers to communicate together, they need to form a ROS network with one common ROS Master. Implementation-wise, the main differences occur in the Franka State Service and the way the information about the state of the robot arm is being gathered from the hardware.

The main differences in the states visited during the door opening procedure are following:

- **Starting state:** It performs the same object initialization as in the Gazebo simulation but it no longer virtually sets the end effector and stiffness frames. Now, through built-in ROS services of the *franka_control* package, it defines the frames to be later used when receiving the information about the sate of the robotic arm. Before the gripper is closed, a homing procedure has to be completed. This is also performed in this particular state. More importantly, we no longer need a connection to the PyBullet library as the information about the arm state will be available through Franka's specialized **libfranka** library. The closing command remains the same as in the Gazebo simulation.
- **Handle Grasped State:** Since we were not performing the actual door opening procedure in the Gazebo simulation (just the simplified tests), there

was no need to implement the initial direction estimation module. On the real robot, the initial direction estimation procedure is performed in the Handle Grasped State. Upon completion, the obtained estimate will be used to initialize the online direction estimation procedure and the state will once again wait for the user to initiate the automatic control procedure.

- **Automatic Control State:** This state no longer collects the information by parsing the URDF model of the Franka arm using the PyBullet library. The Franka State server uses the franka::ReadOnce( ) command from the **libfranka** library to get all the necessary joint information from the on-board sensors. As previously mentioned, **libfranka** provides built-in functions to parse the URDF model of the Franka arm in order to calculate the dynamic parameters for the current robot configuration.
- **Stopping state:** The stopping state releases the grasped object at the end of the run.

A complete list of ROS topics and services used on the real robot is presented in the Table 6.2.

Table 6.2: ROS topics and services used on the real robot.

ROS topics and services	Description
/panda_state_srv	receives the full state of the robot arm
/robot_gripper_srv	closes the gripper and sets its frames
/franka_control_set_EE_frame	defines the transformation matrix $T_{b,EE}$
/franka_control_set_K_frame	defines the transformation matrix $T_{EE,K}$
/ridgeback_velocity_controller/odom	receives $T_{O,base}$ and v_b^{base}
/arm_command	publishes desired joint velocities
/cmd_vel	publishes desired mobile base velocity

Having thoroughly examined the whole body control procedure and the structure of the system that complies with the ROS Control framework, we proceed to first analyse the performance of our system in the simulations and then in the real world. In the next chapter, based on the appropriate metrics, we will investigate how the change in certain hyper-parameters affects the performance of the system. We then go on to present the results obtained both in the simulation and in the real world for the most promising configurations.

Chapter 7

Simulation Experiments

In this chapter we present the experimental results obtained in the two simulated environments with the model of the real Franka Emika robot. Since we use the PyBullet simulation to validate our concept, we start by analysing how different hyper-parameters influence the performance of the complete closed loop control method. The performance for each door model is evaluated by individually varying the hyper-parameters and performing multiple runs with different initial configurations of the robot. The parameters that we do not fix a priori satisfy:

- $f_c \in \{0.5f_0, f_0, 1.5f_0\}$ where f_0 is the nominal cut-off frequency of the digital filter given by $f_0 = \frac{1}{50\Delta T}$
- $c_1/c_2 \in \{0.5, 1.0, 2.0\}$ so that we can demonstrate what happens if different penalisation ratios between the two components of the cost, given by equations 5.16 and 5.30, are used
- $\beta \in \{0.25, 0.5, 0.75\}$ so that we can test how the system performs when different maximal Euclidean norms of the relative velocity between the arm and the base are imposed
- $m \in \{0.25, 0.5, 0.75\}$ in order to describe how the system performs when one of the two unconstrained direction of motion estimates is more dominant

In order to compare the models, we define a set of metrics that properly describes the desired behaviour of our system. We are mainly interested in the manipulability index of the robot, the planning time of the optimizer, how well the online direction estimation performs and the repeatibility of the proposed solution.

Let us denote with N_{runs} the total number of runs performed for a particular configuration of the whole body control algorithm. To evaluate the performance of the online direction estimation over the whole run, we introduce the average dot product between the true and the estimated unconstrained direction of motion:

$$\langle n_{\text{true}}\, , n_{\text{est}} \rangle^{\text{avr}} = \frac{1}{N_{\text{runs}}} \cdot \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{runs}}} \sum_{j=1}^{N_{\text{total}}} \left\langle n^i_{\text{true}}\,[j], n^i_{\text{est}}\,[j] \right\rangle \tag{7.1}$$

We are also interested in evaluating how the online direction estimation performs during the initial period of the procedure. Since we test all the configurations with the same length of the initial period, this metric should roughly describe how fast the direction estimator converges. For this we calculate the average root mean squared error (RMSE):

$$\mathrm{RMSE}_{\mathrm{init}} = \frac{1}{N_{\mathrm{runs}}} \sum_{i=1}^{N_{\mathrm{runs}}} \sqrt{\frac{1}{N_{\text{start-up}}} \sum_{j=1}^{N_{\text{start-up}}} \left(1 - \left\langle n^i_{\mathrm{true}}\,[j], n^i_{\mathrm{est}}\,[j] \right\rangle\right)^2} \tag{7.2}$$

where we set the desired value of the dot product to be equal to 1. Two metrics are defined to evaluate the manipulability aspect of our algorithm. To describe the performance over the whole run, we define the average manipulability index:

$$\mu^{\mathrm{avr}} = \frac{1}{N_{\mathrm{runs}}} \cdot \frac{1}{N_{\mathrm{total}}} \sum_{i=1}^{N_{\mathrm{runs}}} \sum_{j=1}^{N_{\mathrm{total}}} \sqrt{\det\left(J^i_{b,EE}[j] \left(J^i_{b,EE}[j]\right)^T\right)} \tag{7.3}$$

Since one of our main goals is to have high manipulability index towards the end of the door opening procedure, we also report the average manipulability index in the last $N_{\mathrm{last}} = 20$ loop iterations defined as:

$$\mu^{\mathrm{end}} = \frac{1}{N_{\mathrm{runs}}} \cdot \frac{1}{N_{\mathrm{last}}} \sum_{i=1}^{N_{\mathrm{runs}}} \sum_{j=N_{\mathrm{total}}-N_{\mathrm{last}}+1}^{N_{\mathrm{total}}} \sqrt{\det\left(J^i_{b,EE}[j] \left(J^i_{b,EE}[j]\right)^T\right)} \tag{7.4}$$

Finally, we also measure the average total planning time and the average planning time per iteration given by:

$$T^{\mathrm{avr}}_{\mathrm{total}} = \frac{1}{N_{\mathrm{runs}}} \sum_{i=1}^{N_{\mathrm{runs}}} \sum_{j=1}^{N_{\mathrm{total}}} \Delta t_i[j] \tag{7.5}$$

$$\Delta t^{\mathrm{avr}} = \frac{1}{N_{\mathrm{runs}}} \cdot \frac{1}{N_{\mathrm{total}}} \sum_{i=1}^{N_{\mathrm{runs}}} \sum_{j=1}^{N_{\mathrm{total}}} \Delta t_i[j] \tag{7.6}$$

All the simulations were performed on a computer with an Intel i-10750H configuration with 6 CPUs operating in the range from 2.6 to 5GHz.

7.1 PyBullet Simulation Results

The main goal of this chapter is to provide an analysis of the system's performance when deployed with different combinations of the hyper-parameters and the optimization problem formulations. As a baseline model of the system, we choose the fixed base algorithm that has no velocity splitting module but plans the joint velocity commands within the hardware constraints of the robot. By comparing the performance of this model with the ones introduced in Sections 5.3.1, 5.3.2 and 5.3.3, we want to infer the benefits and drawbacks of utilizing the additional DOFs provided by the mobile base.

Let us define a nominal parameter configuration of the system. In all of the cases, we set the length of the end effector position buffer to be equal to $N_{\text{start-up}}$. With regard to the unconstrained direction of motion estimator module, the nominal configuration is given by:

$$m = 0.5; \quad f_c = f_0;$$

where this value of mixing coefficient m enforces equal importance of haptic based and fixed-grasp based estimates. The cut-off frequency of the Butterworth filter was chosen by trial and error procedure such that, at the end, a smooth opening procedure was observed in the simulation. Regarding the velocity splitting module,

the nominal configuration equally penalizes both terms in the cost function by choosing:

$$c_1 = c_2 = 1$$

The parameter β is chosen as 0.5. When we test the performance on the drawer, the room door and the sliding door, we set $N_{\text{total}} = 1100$ whereas in the case of the sliding lid and the dishwasher, we set $N_{\text{total}} = 600$. The dimensions of the sliding lid mechanism impose a shorter opening procedure whereas the position of the dishwasher door handle is no longer in the reachable space of the robotic arm (in vertical direction) after 600 iterations. The α parameter of the unconstrained direction of motion estimator is set to 0.1. The velocities during the initial period and afterwards are given by:

$$v_1 = 0.025\,\text{m/s}; \quad v_2 = 0.1\,\text{m/s};$$

The remaining velocity time profile parameters are set to:

$$T_{\text{init}} = N_{\text{start-up}}; \quad t_0 = \left\lfloor \frac{T_{\text{init}}}{3} \right\rfloor; \quad \alpha_{\text{init}} = \alpha_{\text{regular}} = 0.5;$$

The scaling coefficient for the angular velocity planning is $k = 0.2$ whereas the velocity splitting module paramaters are given by:

$$\alpha = 0.1; \quad R_{\text{lim}} = 0.1; \quad N_p = 32;$$

Every parameter combination that we test is run multiple times. We define a set of different initial positions and orientations of the base that determines the total number of runs to be executed for a particular system configuration. When comparing different mobile base algorithms, each of them starts with the same initial joint configuration that the baseline model had for that particular run. Due to the size of the doors and the mounting positions of the handles, not all initial configurations of the base are always feasible so the total number of runs differs when opening different mechanisms. For the dishwasher door, the number of runs is the smallest and equals 10. For the drawer model, the number of runs is 20 whereas the number of runs for all other types of mechanisms is set to 25. In each of them, the initial grasping orientation is defined by the angular displacement in the horizontal plane between the z axis of the end effector frame and the door normal that is equal to 45°.

For convenience, let us denote with $M = \{m_1, m_2, \dots m_5\}$ the set of all mechanisms such that m_1 corresponds to the dishwasher door, m_2 to the drawer, m_3 to the room door, m_4 to the sliding door and m_5 to the sliding lid mechanism. Furthermore, we define the set of all controllers $C = \{c_1, c_2, c_3, c_4\}$ where c_1 corresponds to the baseline fixed base controller, c_2 to the LCQP mobile base formulation, c_3 to the QCCO formulation and c_4 to the SOCP formulation. It is important to note that, in order to make the initial testing conditions same for every run, at this stage, we turn off the initial direction estimation module and use a fixed initial estimate:

$$e_{EE}^0 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

The remaining part of this chapter is organized such that we first choose the most suitable problem formulation for the velocity splitting procedure. We continue by performing thorough analysis of the chosen algorithm for the nominal set of parameters and report its results when combined with the initial direction estimation procedure. Finally, before we proceed to test on the real robot, we do a small hyper-parameter exploration and report the results of the simple trajectory tracking experiments performed in the Gazebo simulation.

7.1.1 Choosing the Velocity Splitting Problem Formulation

We have proposed three different velocity splitting problem formulations that rely on different third party optimization libraries. In order to choose the best one, we report the performance of each controller for all 5 door models. First of all, it is to be expected that they differ in average planning time per iteration. Figure 7.1 shows how the average execution time per iteration behaves for different optimization algorithms.

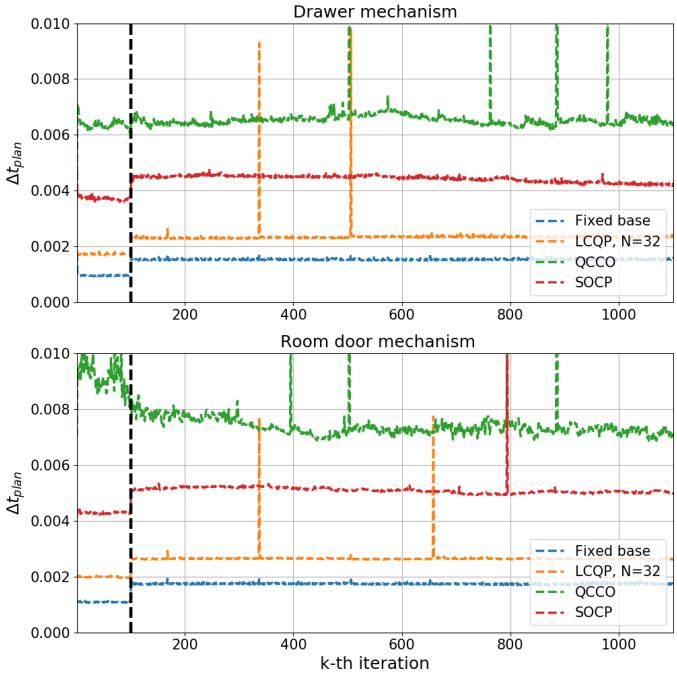


Figure 7.1: Average planning time per iteration when operating the drawer and the room door mechanisms.

The fixed base algorithm is the fastest since it requires one less optimization procedure than the mobile base ones. LCQP formulation is close in terms of execution speed with an average planning time per iteration that is approximately 0.8 ms longer. SOCP and QCCO are approximately 2.5 and 3.5 times slower. Though we would prefer to have as fast an algorithm as possible, all of these execution times are more than suitable to satisfy the real time requirements of our application. Therefore, the average planning time per iteration should not be the deciding factor when picking the best mobile base algorithm.

We proceed by analysing the average value of metrics defined in this section for every door model and every controller. Furthermore, as we are interested in having a mobile base algorithm that can reliably operate different doors, we also report the number of failed runs for each of the controllers. A run is considered to be failed if the end effector slips off the handle or the feasible set of joint velocities becomes empty at a particular loop iteration. Note that the averaging of the metrics is always done by taking into account only the successful runs. Table 7.1 shows the average performance of the fixed base algorithm for different door types. We provide the same table for the three mobile base controllers. Table 7.2 corresponds to the LCQP problem formulation, Table 7.3 corresponds to the QCCO formulation whereas the Table 7.4 presents the results for the SOCP optimization formulation.

Table 7.1: Average metrics for the fixed base controller.

Metric	m_1	m_2	m_3	m_4	m_5
$\mathrm{RMSE}_{\mathrm{init}}$	0.0407	0.0485	0.2144	0.0691	0.0645
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	0.9841	0.9957	0.9720	0.9973	0.9821
μ^{avr}	0.0653	0.0782	0.0541	0.0671	0.0823
μ^{end}	0.0815	0.0437	0.0348	0.0543	0.0781
$T_{\mathrm{total}}^{\mathrm{avr}}$	0.7079	1.6320	1.8446	1.8082	0.7753
Δt^{avr}	0.00142	0.00148	0.00168	0.00164	0.00155
N_{fail}	4	6	10	8	5

Table 7.2: Average metrics for the LCQP controller.

Metric	m_1	m_2	m_3	m_4	m_5
$\mathrm{RMSE}_{\mathrm{init}}$	0.0633	0.0373	0.1212	0.0964	0.0655
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	0.9226	0.9961	0.9844	0.9797	0.9836
μ^{avr}	0.0256	0.0671	0.0567	0.0569	0.0833
μ^{end}	0.0273	0.0738	0.0581	0.0601	0.0819
$T_{\mathrm{total}}^{\mathrm{avr}}$	1.1236	2.2524	2.8452	2.7960	1.2379
Δt^{avr}	0.00224	0.00229	0.00258	0.00254	0.00247
N_{fail}	4	7	6	4	0

Table 7.3: Average metrics for the QCCO controller.

Metric	m_1	m_2	m_3	m_4	m_5
$\mathrm{RMSE}_{\mathrm{init}}$	0.0397	0.0472	0.0738	0.0565	0.0711
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	0.9858	0.9952	0.9888	0.9981	0.9821
μ^{avr}	0.0473	0.0786	0.0615	0.0615	0.0829
μ^{end}	0.0644	0.0863	0.0665	0.0691	0.0811
$T_{\mathrm{total}}^{\mathrm{avr}}$	3.4104	7.2158	8.3277	11.3387	4.1224
Δt^{avr}	0.00682	0.00656	0.00757	0.01031	0.00824
N_{fail}	3	5	6	5	0

Table 7.4: Average metrics for the SOCP controller.

Metric	m_1	m_2	m_3	m_4	m_5
$\mathrm{RMSE}_{\mathrm{init}}$	0.0707	0.0524	0.1399	0.0611	0.0662
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	0.9645	0.9956	0.9806	0.9978	0.9836
μ^{avr}	0.0472	0.0794	0.0634	0.0615	0.0833
μ^{end}	0.0642	0.0869	0.0683	0.0693	0.0820
$T_{\mathrm{total}}^{\mathrm{avr}}$	2.2058	4.8077	5.5201	5.4041	2.3238
Δt^{avr}	0.00441	0.00437	0.00502	0.00491	0.00465
N_{fail}	3	5	6	5	0

These tables show the average performance of the models for each individual door type. They show that the controllers perform better on certain types of articulated models compared to the others. In order to jointly analyse the performance for all types of door mechanisms, we average the values of the metrics over the set of all door types. Furthermore, to algebraically describe the reliability of a controller, we define its expected number of failed runs when operating any type of door mechanisms. If for a given controller c we denote with $N_{m_1}, N_{m_2}, \ldots, N_{m_5}$ the number of failed runs for the corresponding door type, we define the expected number of failed runs as:

$$\hat{N}_{\text{fail}} = \frac{10}{105} N_{m_1} + \frac{20}{105} N_{m_2} + \frac{25}{105} N_{m_3} + \frac{25}{105} N_{m_4} + \frac{25}{105} N_{m_5} \tag{7.7}$$

The complete performance of the controllers is given in the Table 7.5.

Table 7.5: Average performance of all controllers.

Metric	c_1	c_2	c_3	c_4
$\text{RMSE}_{\text{init}}$	0.0875	0.0768	**0.0576**	0.0781
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9862	0.9733	**0.9900**	0.9844
μ^{avr}	0.06941	0.05793	0.0664	**0.0669**
μ^{end}	0.05851	0.0603	0.0735	**0.0741**
$T^{\text{avr}}_{\text{total}}$	1.3536	**2.1054**	6.8830	4.0523
Δt^{avr}	0.00155	**0.00243**	0.00790	0.00467
$\hat{N}_{\text{fail}}$	7	4.0952	**3.8571**	**3.8571**

In terms of reliability, we see that the QCCO and the SOCP formulations perform best since they have the smallest expected number of failed runs. On the other hand, regarding the average planning time per iteration, these two controllers are on the other side of the spectrum. With regards to the end effector manipulability, we see that QCCO and SOCP significantly outperform the fixed base algorithm and the LCQP. Finally, the QCCO achieves the smallest average RMSE of the dot product between the true and the estimated unconstrained direction of motion during the initial period of the door opening procedure. What is more, it also outperforms the other controllers in the average dot product over the whole run though the difference is not that big in this case. With all this in mind, we choose the quadratically constrained convex optimization formulation for the velocity splitting procedure. Although it is the slowest proposed solution, it provides the highest reliability, achieves the best unconstrained direction of motion estimation and performs approximately the same as the SOCP algorithm in terms of the average manipulability index at the end of the run. It is important to note that the SOCP controller can also be considered a viable solution for our task.

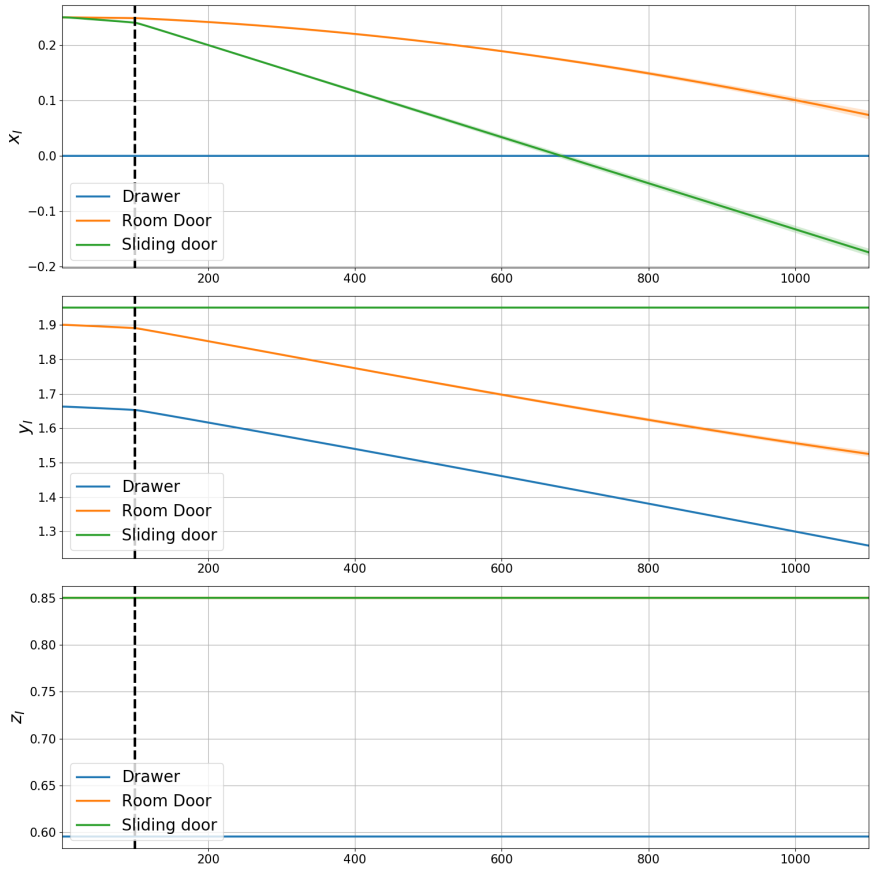
7.1.2 Performance of the Chosen Controller

In this subsection we give thorough performance analysis for the QCCO controller with the nominal set of parameters. In all of the testing scenarios, the door mechanisms were oriented in the environment such that the drawer is always opened in the negative y direction of the world frame. The sliding door is opened in the negative x direction of the world frame whereas the sliding lid is always opened in the x direction of the world frame. The room door model is always moving in the x-y plane of the world frame such that during the entire procedure the door handle has velocity components in the negative x and the negative y direction of the world frame. The dishwasher door moves in the plane perpendicular to the horizontal

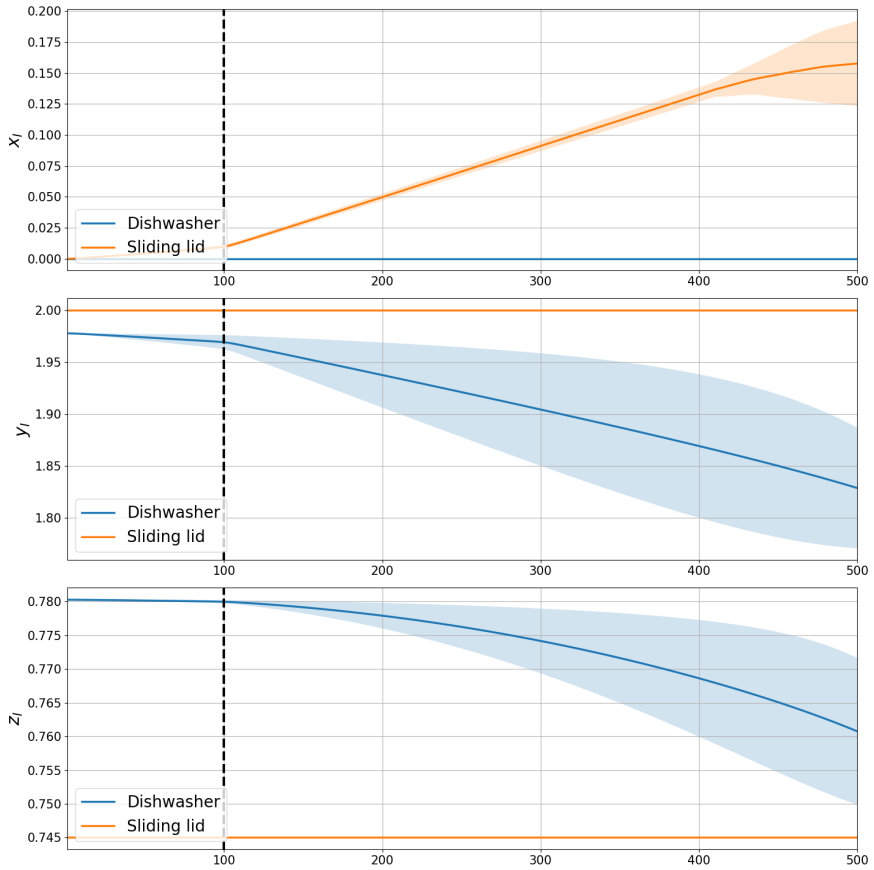
ground plane such that the handle always has a velocity component in the negative y and the negative z direction of the world frame. Note that from this point on, every averaged plot that we provide will also include the $\pm 2\sigma$ interval to indicate the uncertainty about the particular measurement.

Figure 7.2 shows the averaged end effector position when operating each of the door models. Since we rely on a fixed grasp during the entire run, these two plots effectively give the position of the end effector in the world frame. Figures 7.3 and 7.4 show the performance comparison between the fixed base and the mobile base algorithms for all types of doors. Due to the nature of the door opening task, after certain amount of time, the fixed base algorithm has to start suffering from a significant decrease in manipulability. This is not the case for the mobile base. In the worst case scenario, it can preserve most of its manipulabilty by achieving most of the desired movement through the movement of the mobile base. Hence, in all of the cases the manipulability index either negligibly decreases (if the robot starts from a relaxed initial configuration) or increases over time. When it comes to the average angular displacement between the true and the estimated unconstrained direction of motion, for the drawer, the room door and the sliding door models, the $\pm 2\sigma$ interval can be upper bounded by 10° after approximately 400 iterations for both the mobile and the fixed base algorithms. This is not the case for the dishwasher door and the sliding lid as here the upper bound for the dishwasher door has to be increased to 20°, indicating worse performance in terms of unconstrained direction of motion estimation. The upper bound on the average planning time per iteration is considerably smaller in the fixed base scenario (approx. 2 ms) than in the mobile base scenario (approx. 12.5 ms). There is no significant difference in average planning time per iteration coming from different door types except when operating the sliding door using the mobile base algorithm.

Finally, Figure 7.5 shows the end effector velocity in the world frame for the two door mechanisms that will also be operated using the real robot. Solid lines in the plots correspond to the measured values whereas the dotted lines in the v_x, v_y and v_z subplots indicate the relative velocity between the end effector and the base obtained as one of the outputs of the velocity splitting optimization procedure. As can be seen from Figures 7.5(a) and 7.5(b), the measured absolute value of the end effector velocity matches the one sent by the velocity time profile module. In the drawer case, the velocity is mainly in the y direction of the world frame whereas in the room door case, there exists also a component in the x direction of the world frame.

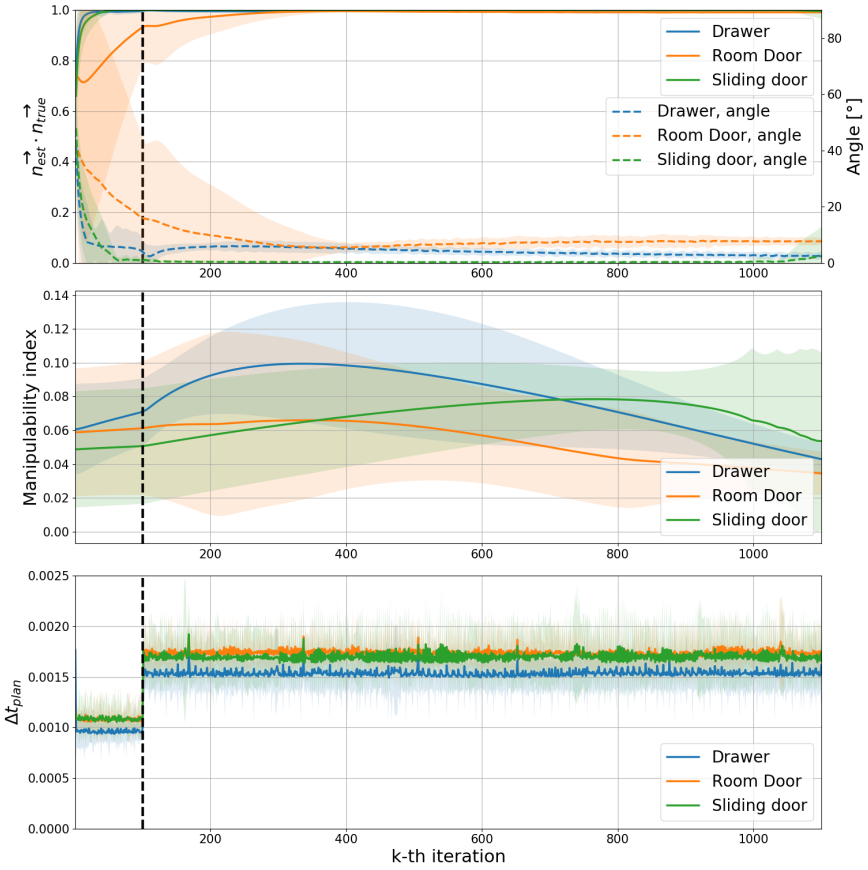


(a) Drawer, room door and sliding door mechanisms

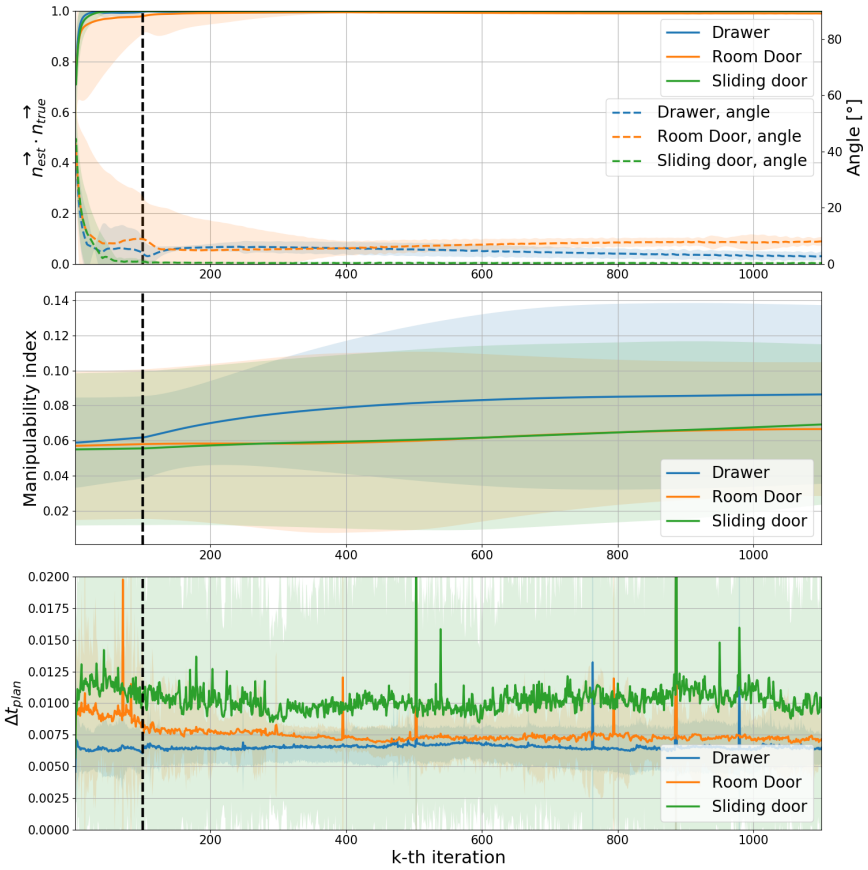


(b) Dishwasher and sliding lid mechanisms

Figure 7.2: End effector position when QCCO is deployed.

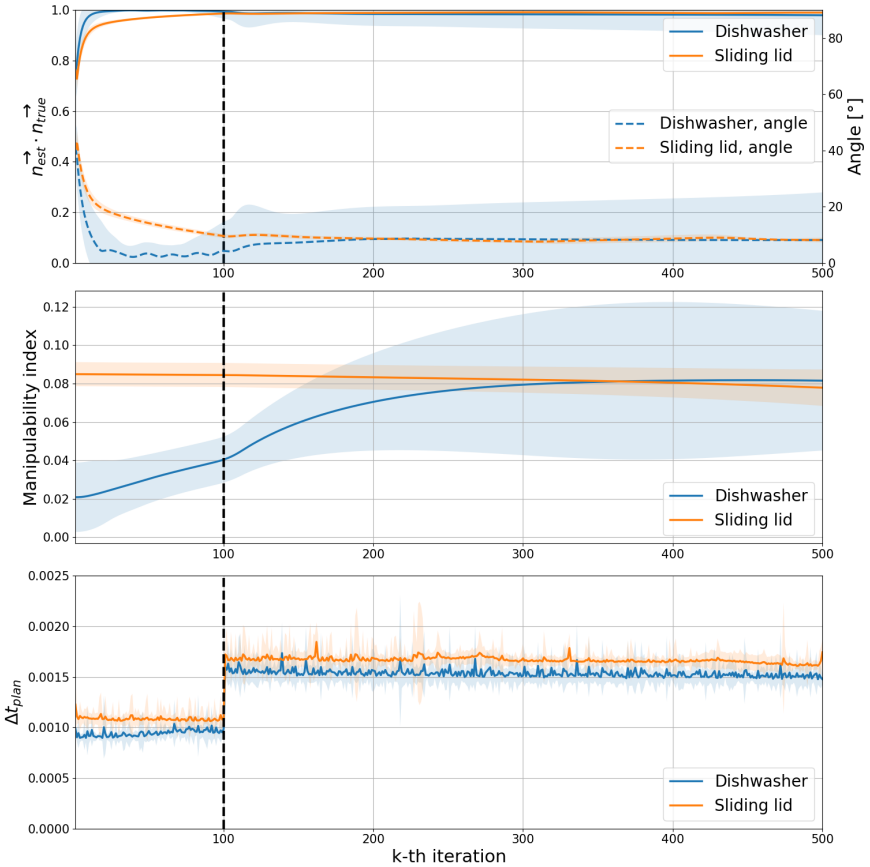


(a) Fixed base

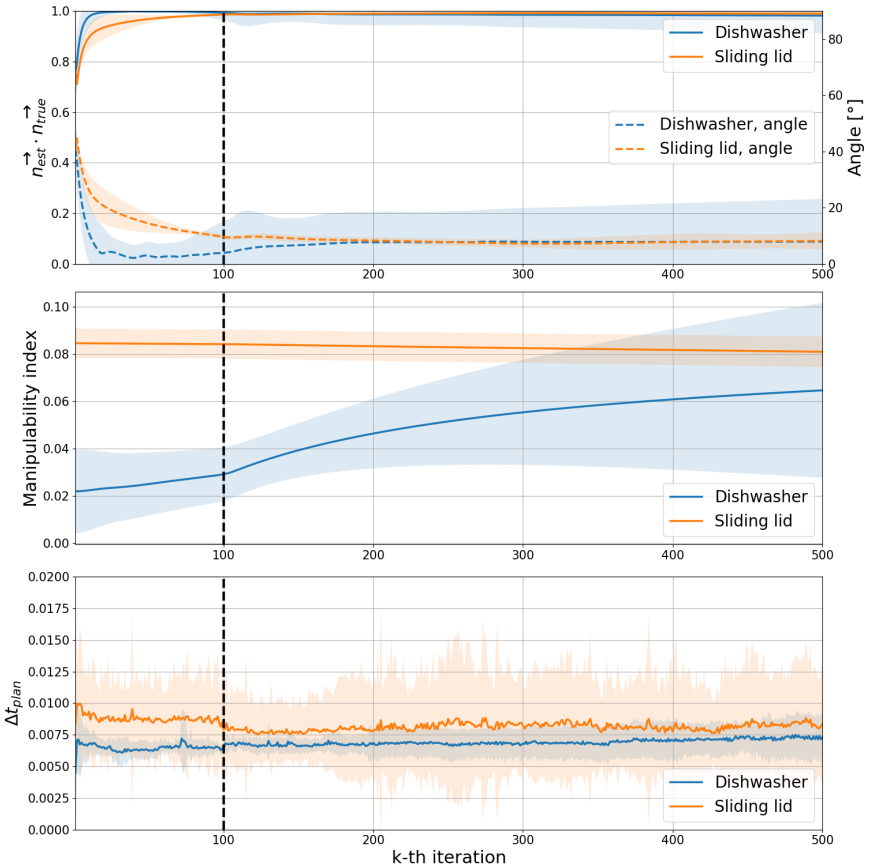


(b) Mobile base system

Figure 7.3: System performance when operating drawer, room door and sliding door models.

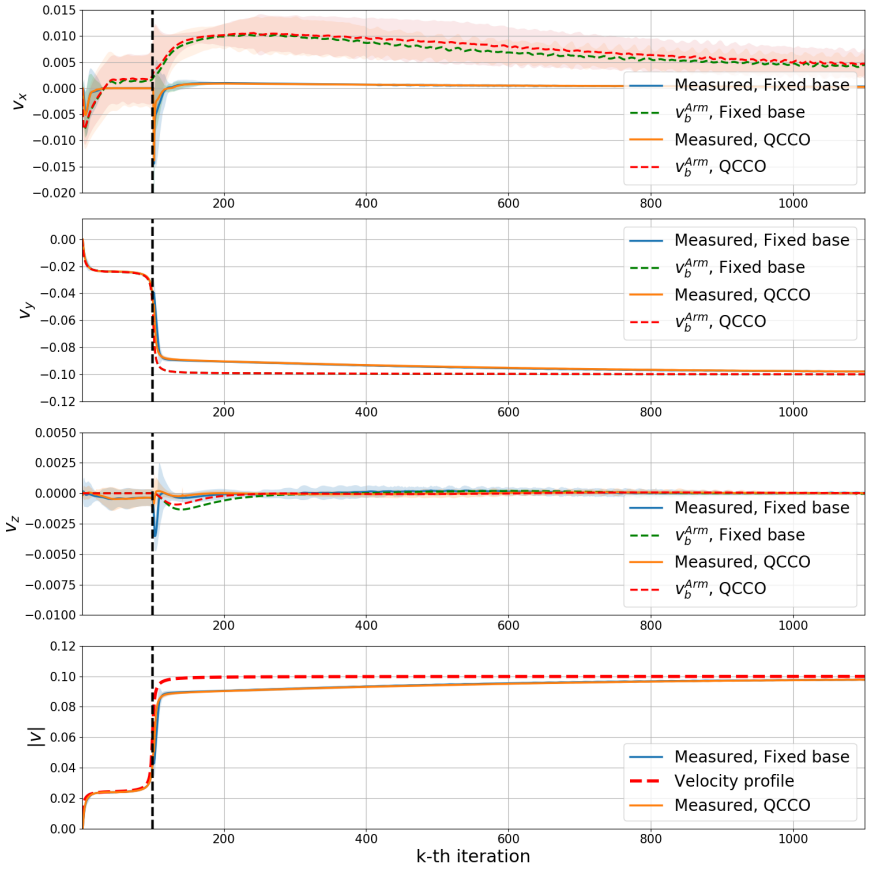


(a) Fixed base

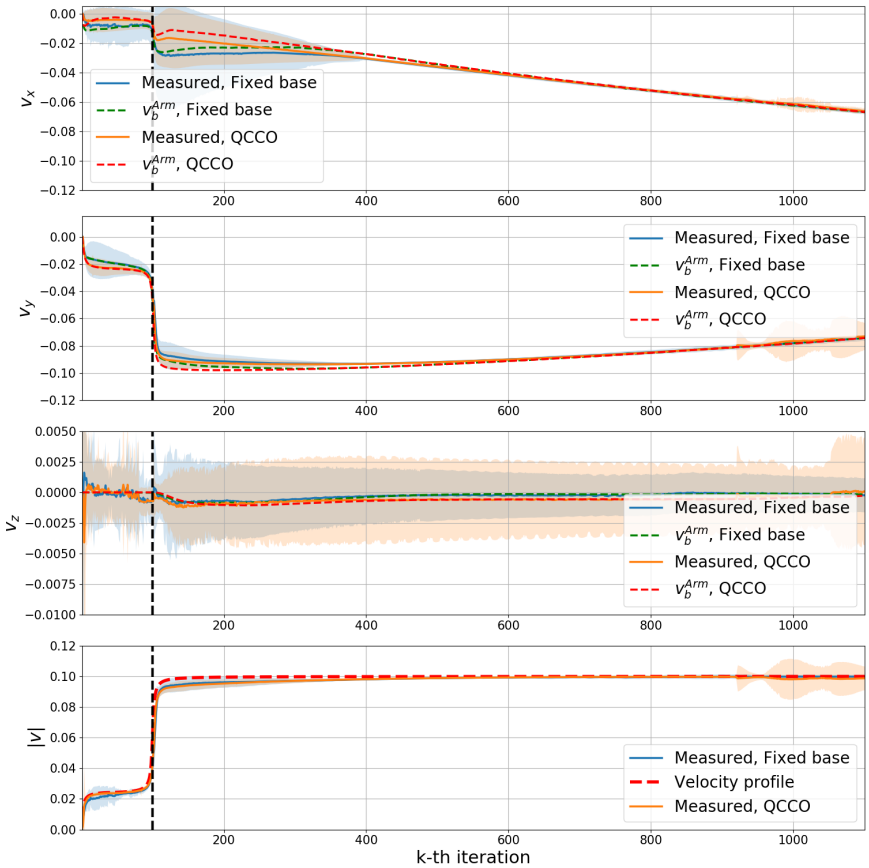


(b) Mobile base system

Figure 7.4: System performance when operating dishwasher door and sliding lid.



(a) Velocity when opening a drawer



(b) Velocity when opening a room door

Figure 7.5: End effector velocity plots.

7.1.3 Simulation With the Initial Direction Estimation

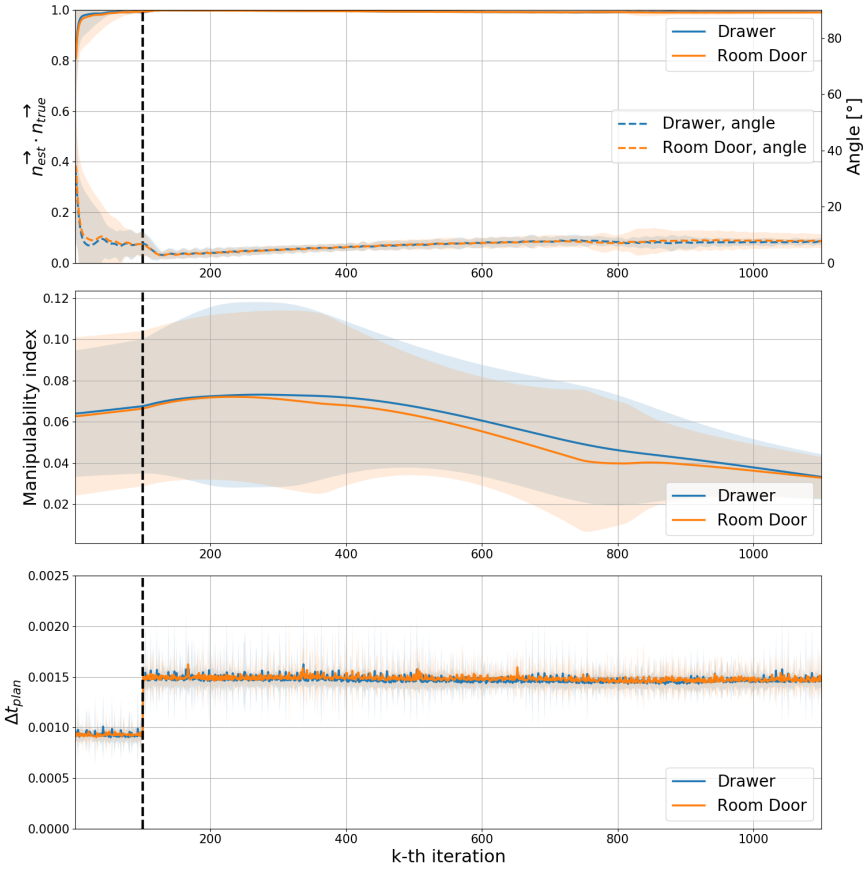
In the previous subsections, we thoroughly tested the chosen mobile base algorithm using a fixed initial direction estimate. To round up the testing phase of the model with the nominal parameters, we also deploy the initial direction estimation procedure. Since we are going to test our algorithm on the real robot when operating the drawer and the cabinet door models, from this point on, we analyse the performance of our models only for these two door types. Table 7.6 and Figure 7.6 summarize the results of all successfully completed runs.

Table 7.6: Complete algorithm performance.

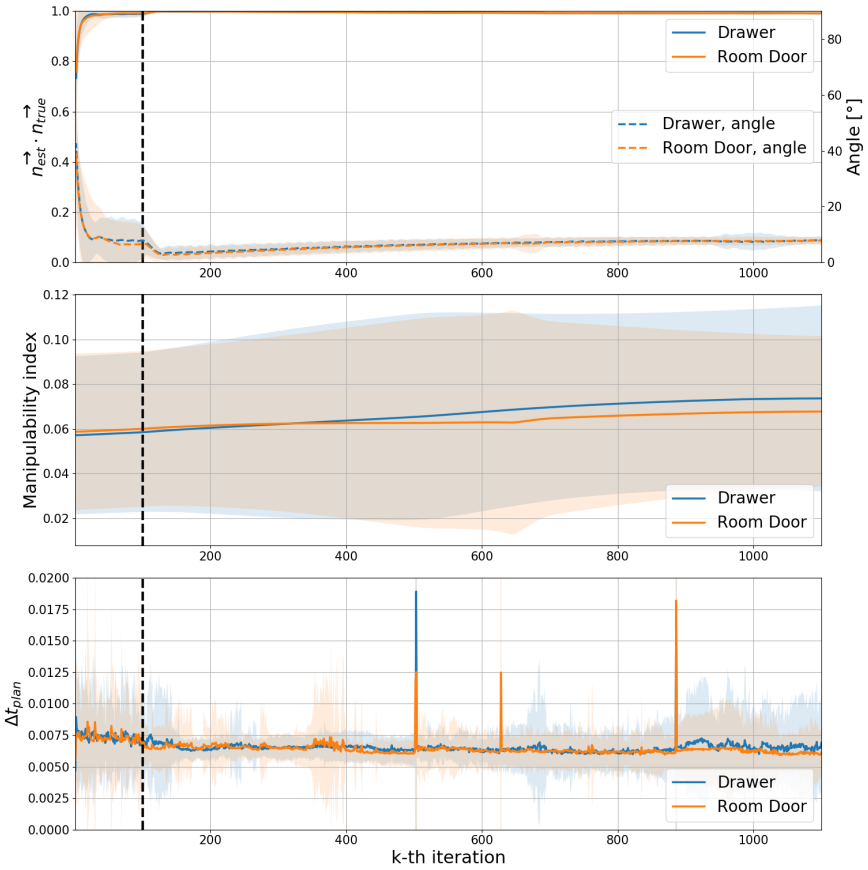
Metric	Drawer		Room door	
	Fixed base	Mobile base	Fixed base	Mobile base
$\text{RMSE}_{\text{init}}$	**0.0373**	0.0503	**0.0442**	0.0505
$\langle n_{\text{true}}\ , n_{\text{est}}\ \rangle^{\text{avr}}$	**0.9925**	0.9916	0.9917	**0.9919**
μ^{avr}	0.0583	**0.0664**	0.0550	**0.0637**
μ^{end}	0.0337	**0.0736**	0.0332	**0.0678**
$T^{\text{avr}}_{\text{total}}$	**1.5600**	7.2376	**1.5751**	7.1069
Δt^{avr}	**0.00142**	0.00658	**0.00143**	0.00646

It can be observed that the increase in the average manipulability index at the end of the run, when the mobile base is utilized, is significant for both types of door mechanisms. This is also the case for the average manipulability index over the whole run, though the relative difference is smaller. The RMSE of the dot product between the true and the estimated unconstrained direction of movement indicates better performance for the fixed base algorithm though, over the whole run, the performance is almost identical. By looking at the Figure 7.6, we can observe that for both algorithms, the $\pm 2\sigma$ interval of the angular displacement between the true and the estimated unconstrained direction of motion can be upper bounded by 10° after approximately 150 loop iterations.

With this subsection we complete the analysis of the fixed base and the mobile base algorithms for the nominal set of model parameters. In the following section, we individually vary the m, β, f_c, c_1 and c_2 hyper-parameters and report the observed changes in performance. This is important as it might provide us with some intuition that could be useful for the potential parameter tuning on the real robot.



(a) Fixed base



(b) Mobile base

Figure 7.6: Performance plots for the runs with the initial direction estimation procedure.

7.1.4 Hyper Parameter Variation

Now that we have thoroughly examined the fixed base and the chosen mobile base algorithm with the nominal set of parameters, the main goal of this chapter would be to analyse what happens if we individually change the value of some parameters.

Varying Parameter m

We start by analysing the mixing coefficient parameter m whose nominal value is set to 0.5. Tables 7.7 and 7.8 show the metrics when operating the drawer and the room door mechanisms, averaged over the set of all successful runs. Figure 7.7 shows the corresponding performance plots.

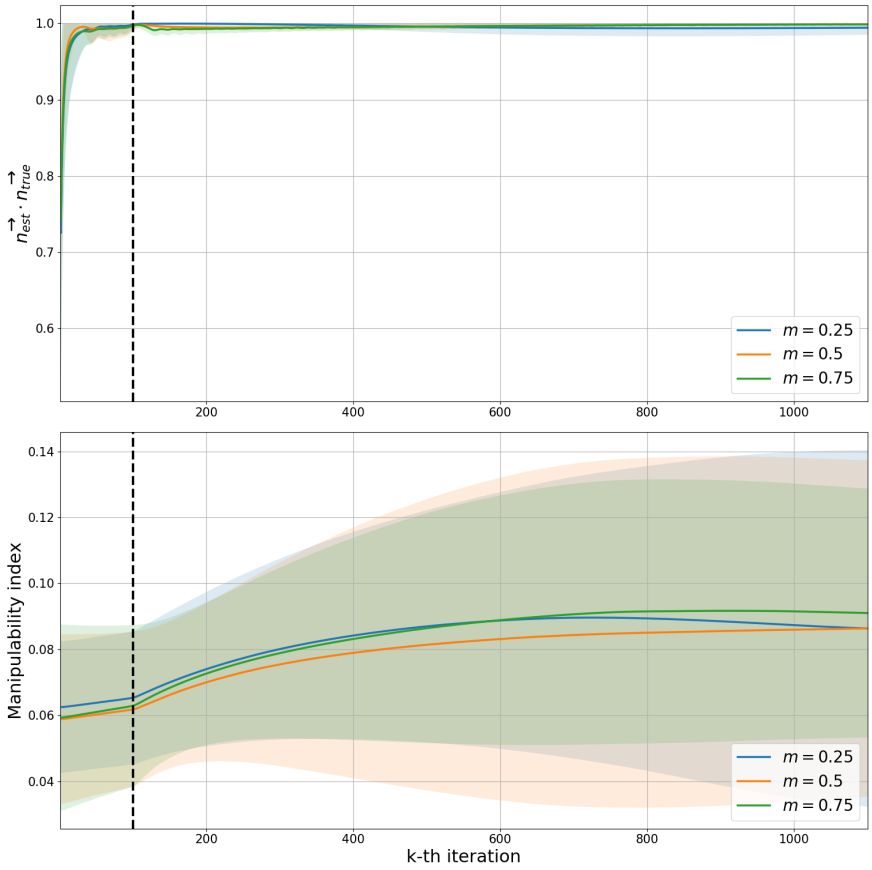
Table 7.7: Varying the parameter m, drawer mechanism.

Metric	$m = 0.25$	$m = 0.5$	$m = 0.75$
$\text{RMSE}_{\text{init}}$	0.0546	**0.0472**	0.0515
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9941	**0.9953**	0.9945
μ^{avr}	0.0827	0.0786	**0.0831**
μ^{end}	0.0864	0.0863	**0.0911**
$T^{\text{avr}}_{\text{total}}$	8.1283	**7.2162**	8.0308
Δt^{avr}	0.00739	**0.00656**	0.00730
N_{fail}	9	**5**	7

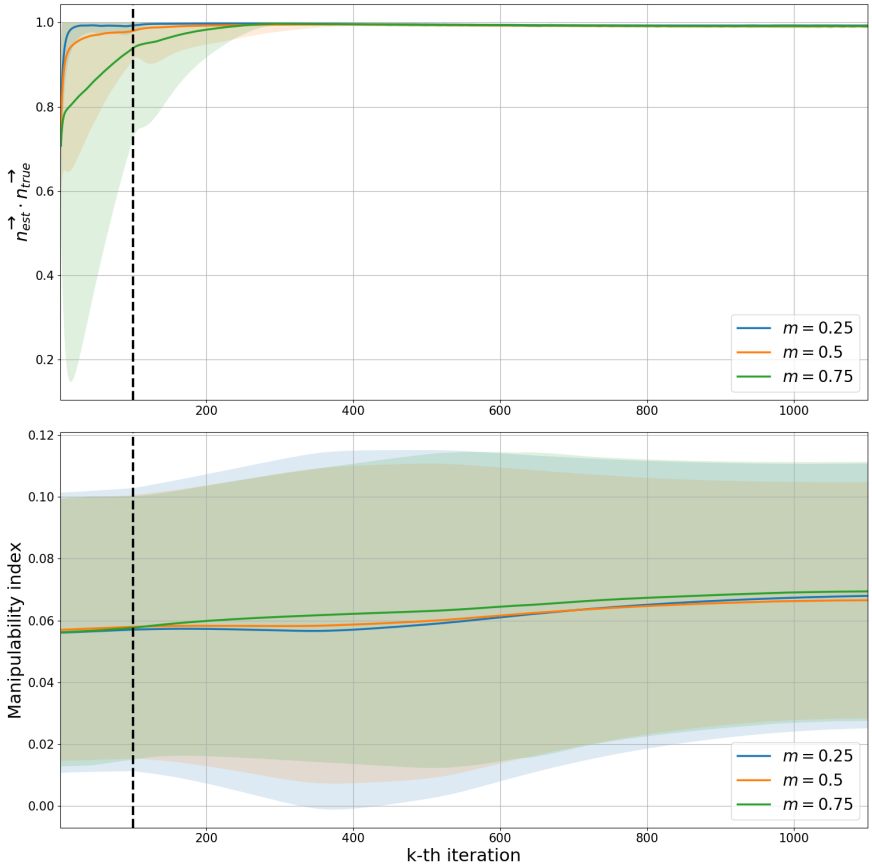
Table 7.8: Varying the parameter m, door mechanism.

Metric	$m = 0.25$	$m = 0.5$	$m = 0.75$
$\text{RMSE}_{\text{init}}$	**0.0457**	0.0738	0.1679
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	**0.9931**	0.9888	0.9782
μ^{avr}	0.0611	0.0615	**0.0638**
μ^{end}	0.0679	0.0665	**0.0694**
$T^{\text{avr}}_{\text{total}}$	**8.0781**	8.3277	8.3139
Δt^{avr}	**0.00734**	0.00757	0.00756
N_{fail}	4	6	**3**

By decreasing the value of parameter m, we effectively favour the fixed-grasp based direction estimate more than the haptic based one and vice versa. Furthermore, larger values of the mixing coefficient make the system more responsive to the sudden changes in the external forces acting on the robot. By examining the two tables, we see that the value $m = 0.75$ achieves the smallest combined number of failed runs. Regarding both manipulability metrics, this configuration also outperforms the others for both door models. Important to note here is that there is no point in comparing the average RMSE of the dot product during the initial period as only the haptic feedback is turned on. Regarding the average dot product over the whole run, the value $m = 0.5$ insignificantly outperforms the others for the drawer mechanism case whereas for the door mechanism, the value $m = 0.25$ actually outperforms $m = 0.5$. With all this in mind, the value of $m = 0.75$ seems like the best candidate as it outperforms the other parameter choices in most cases. However, we have to bear in mind that these differences are not drastic so there in no reason to believe that other choices would lead to a significant degradation in system performance when deployed on the real robot.



(a) Drawer mechanism



(b) Door mechanism

Figure 7.7: Performance for different values of parameter m.

Varying Parameter f

The cut off frequency is the second parameter that describes the online direction estimation procedure. Tables 7.9 and 7.10 show the metrics when operating the drawer and the room door mechanisms, averaged over the set of all successful runs. Figure 7.8 shows the relevant corresponding performance plots for the drawer and the door mechanism.

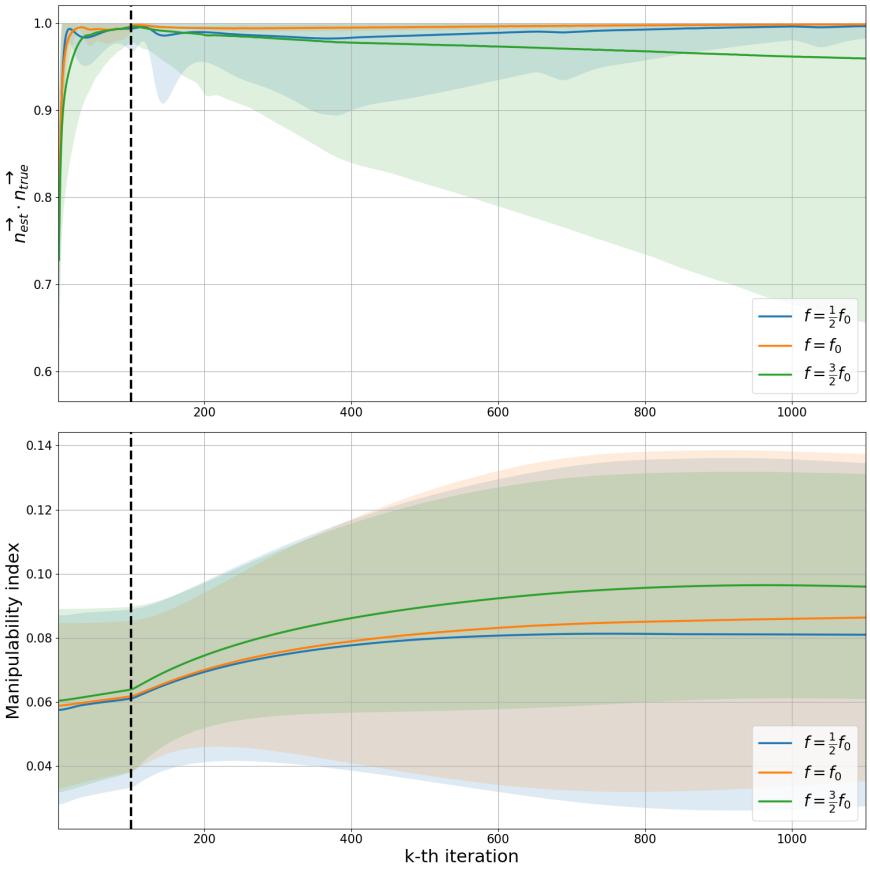
Table 7.9: Varying the parameter f, drawer mechanism.

Metric	$f = 0.5f_0$	$f = f_0$	$f = 1.5f_0$
$\text{RMSE}_{\text{init}}$	0.0518	**0.0472**	0.0615
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9891	**0.9953**	0.9735
μ^{avr}	0.0762	0.0786	**0.0864**
μ^{end}	0.0810	0.0863	**0.0960**
$T^{\text{avr}}_{\text{total}}$	7.9402	**7.2160**	8.3744
Δt^{avr}	0.00722	**0.00656**	0.00761
N_{fail}	6	5	**4**

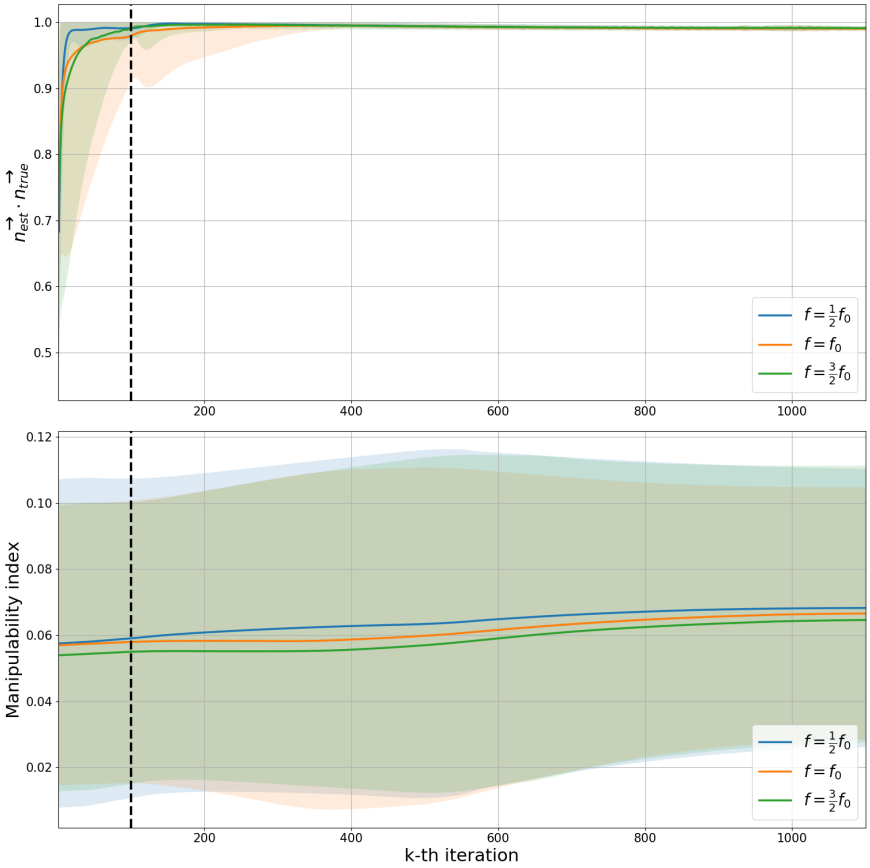
Table 7.10: Varying the parameter f, door mechanism.

Metric	$f = 0.5f_0$	$f = f_0$	$f = 1.5f_0$
$\text{RMSE}_{\text{init}}$	**0.0610**	0.0738	0.0767
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	**0.9919**	0.9888	0.9900
μ^{avr}	**0.0641**	0.0615	0.0589
μ^{end}	**0.0682**	0.0665	0.0646
$T^{\text{avr}}_{\text{total}}$	**8.0235**	8.3277	8.1928
Δt^{avr}	**0.00729**	0.00757	0.00745
N_{fail}	7	6	**4**

As previously stated, the nominal value of the cut off frequency was manually tuned until a smooth door opening procedure was observed in the simulator. Higher values of the cut off frequency allow for a faster response of the system. This is particularly desirable when operating rotational joint mechanisms where the actual unconstrained direction of motion is constantly changing. Having said that, it comes as no surprise that the parameter value $f_c = 1.5f_0$ achieves the smallest number of failed runs. However, higher cut off frequency values could also lead to a more oscillatory character of the estimated outputs that might, in the long run, cause a decrease in observed performance while still providing a successful completion of the opening procedure. This is shown in Figure 7.8(a), where the plots for the drawer case are presented. Here, the $\pm 2\sigma$ interval of the dot product between the true and the estimated unconstrained direction is significantly wider towards the end of the run for $f_c = 1.5f_0$. In all other segments, all three parameter choices give similar results except for the manipulability index at the end of the run for the drawer case, where the highest value of the cut off frequency achieves the best performance. Taking everything into account, the nominal value of the cut off frequency seems like the best option of the three that we presented. It achieves smaller combined number of failed runs than $f_c = 0.5f_0$, while at the same time achieving better, or at least comparable performance in other segments. The value $1.5f_0$ carried the most uncertainty towards the end of the run, so for the initial implementation on the real robot it should not be the first choice.



(a) Drawer mechanism



(b) Door mechanism

Figure 7.8: Performance plots for different values of parameter f.

Varying Parameter β

The value of the β parameter effectively determines the highest possible magnitude of the relative velocity between the arm and the base allowed for the optimizer. Tables 7.11 and 7.12 show the metrics when operating the drawer and the room door mechanisms, averaged over the set of all successful runs. The relevant corresponding performance plots are presented in the Figure 7.9.

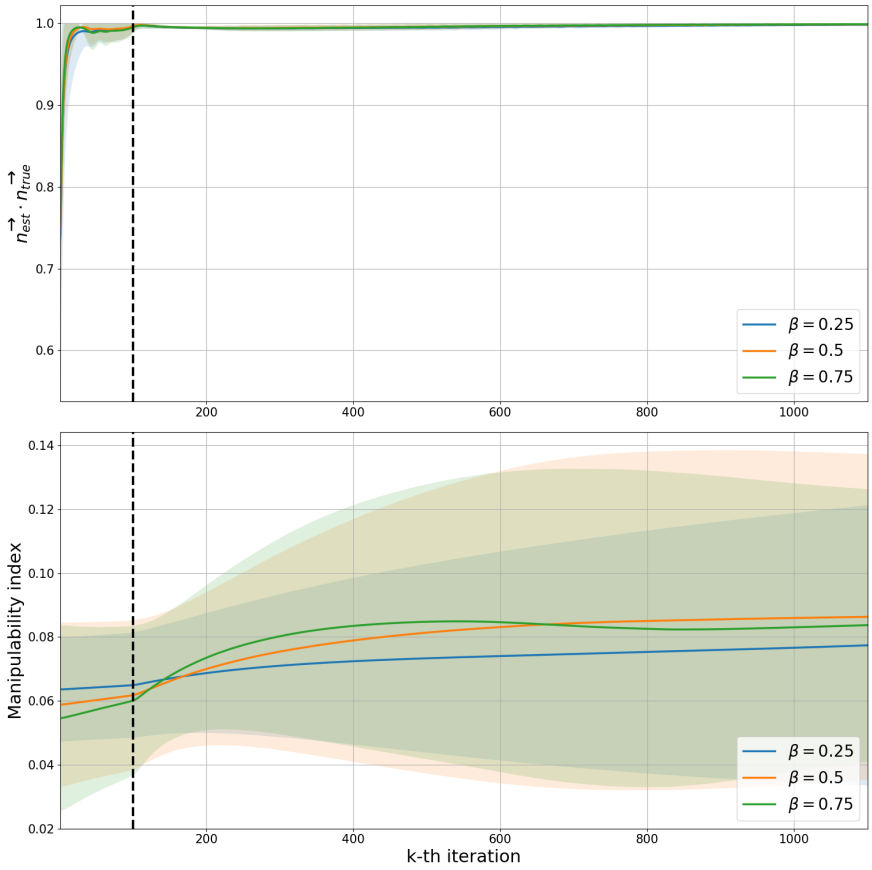
Table 7.11: Varying the parameter β, drawer mechanism.

Metric	$\beta = 0.25$	$\beta = 0.5$	$\beta = 0.75$
$\mathrm{RMSE}_{\mathrm{init}}$	0.0498	0.0472	**0.0428**
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	0.9944	**0.9953**	**0.9953**
μ^{avr}	0.0725	0.0786	**0.0790**
μ^{end}	0.0773	**0.0863**	0.0837
$T_{\mathrm{total}}^{\mathrm{avr}}$	8.8667	**7.2160**	7.6107
Δt^{avr}	0.00806	**0.00656**	0.00692
N_{fail}	**5**	**5**	10

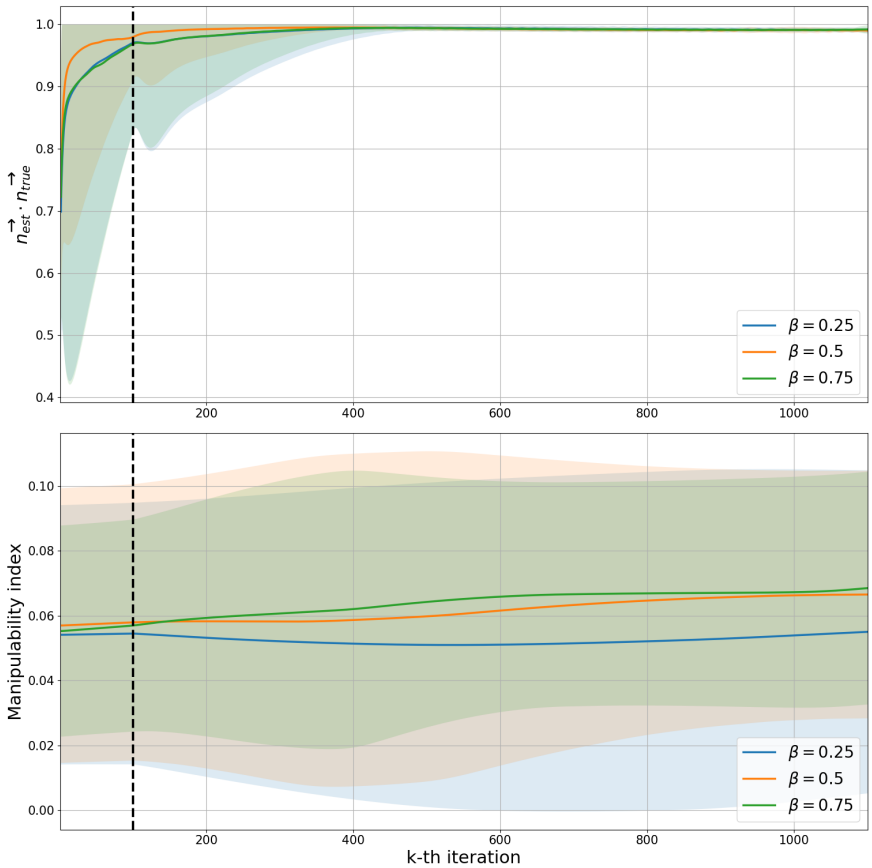
Table 7.12: Varying the parameter β, door mechanism.

Metric	$\beta = 0.25$	$\beta = 0.5$	$\beta = 0.75$
$\mathrm{RMSE}_{\mathrm{init}}$	0.1107	**0.07377**	0.1074
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	0.9837	**0.9888**	0.9836
μ^{avr}	0.0526	0.0615	**0.0636**
μ^{end}	0.0549	0.0665	**0.0684**
$T_{\mathrm{total}}^{\mathrm{avr}}$	8.9572	8.3277	**7.9731**
Δt^{avr}	0.00814	0.00757	**0.00725**
N_{fail}	**6**	**6**	7

Looking at the combined number of the failed runs for the two mechanism types, it is evident that the highest proposed value of the β parameter achieves significantly worse performance. Furthermore, since it is comparable in performance to the other β choices in the rest of the categories, there would be no obvious benefit of choosing this particular value for our mobile base algorithm. When it comes to $\beta = 0.25$ and $\beta = 0.5$, the parameter choice $\beta = 0.5$ outperforms the other in every single category except for the number of failed runs where they achieve the same performance. Therefore, the nominal value of $\beta = 0.5$ is the best option out of the three proposed.



(a) Drawer mechanism



(b) Door mechanism

Figure 7.9: Performance plots for different values of the β parameter.

Varying the Ratio c_1/c_2

The final aspect of the tuning procedure that we take into consideration is the ratio between the two penalization coefficients introduced in the cost function. Tables 7.13 and 7.14 show the metrics when operating the drawer and the room door mechanisms, averaged over the set of all successful runs. The performance plots are presented in Figure 7.10.

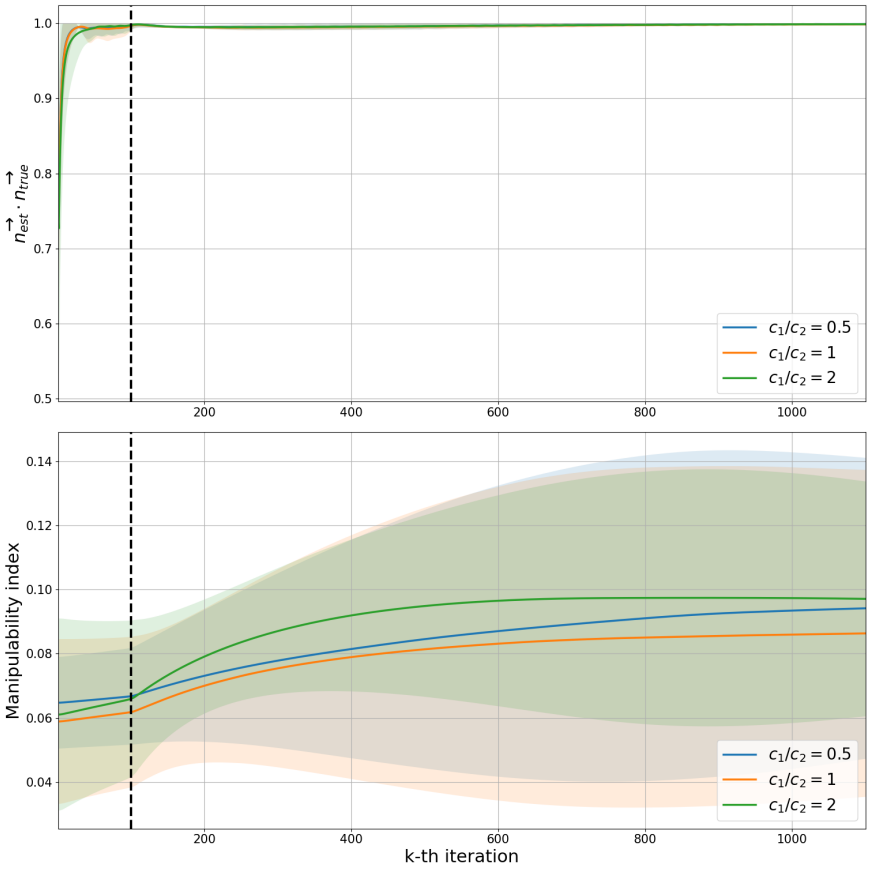
Table 7.13: Varying the ratio c_1/c_2, drawer mechanism.

Metric	$c_1/c_2 = 0.5$	$c_1/c_2 = 1$	$c_1/c_2 = 2$
$\text{RMSE}_{\text{init}}$	**0.0445**	0.0472	0.0541
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	**0.9957**	0.9953	0.9953
μ^{avr}	0.0834	0.0786	**0.0896**
μ^{end}	0.0941	0.0863	**0.0971**
$T^{\text{avr}}_{\text{total}}$	8.7281	7.2160	**6.5662**
Δt^{avr}	0.00793	0.00656	**0.00597**
N_{fail}	11	**5**	**5**

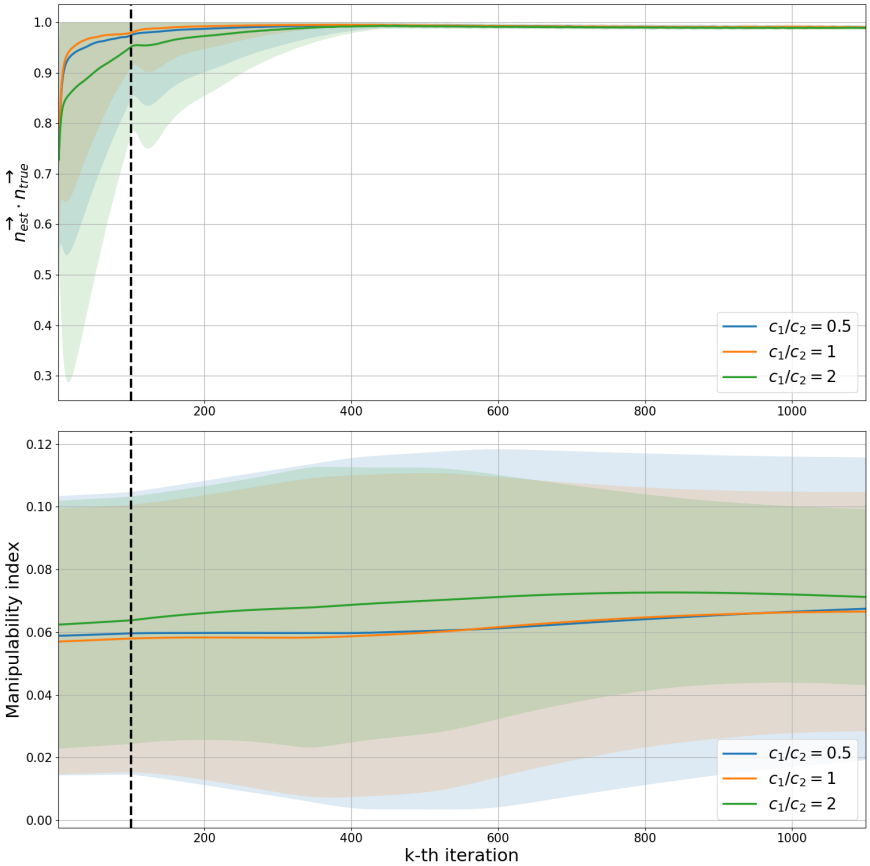
Table 7.14: Varying the ratio c_1/c_2, door mechanism.

Metric	$c_1/c_2 = 0.5$	$c_1/c_2 = 1$	$c_1/c_2 = 2$
$\text{RMSE}_{\text{init}}$	0.08232	**0.07377**	0.1313
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9868	**0.9888**	0.9782
μ^{avr}	0.0620	0.0615	**0.0694**
μ^{end}	0.0674	0.0665	**0.0713**
$T^{\text{avr}}_{\text{total}}$	8.9227	8.3277	**6.6165**
Δt^{avr}	0.00811	0.00757	**0.00601**
N_{fail}	7	**6**	**6**

By setting $c_1/c_2 = 0.5$, we penalize more the term that is supposed to minimize the discrepancy between the relative end effector velocity with respect to the base and the Cartesian velocity that would guide the joints towards the mid point of their position range. In addition, this ratio also allows for worse minimization of the corresponding base velocity. By looking at Tables 7.13 and 7.14, we can see that this leads to the most failed runs for both types of the door mechanism. Ratios $c_1/c_2 = 1$ and $c_1/c_2 = 2$ achieve drastically better repeatibility of the algorithm. In terms of average RMSE of the dot product, the ratio $c_1/c_2 = 1$ significantly outperforms the value $c_1/c_2 = 2$. It also performs no worse in terms of the average dot product over the whole run. On the other hand, $c_1/c_2 = 2$ outperforms the ratio $c_1/c_2 = 1$ in all manipulability related categories, though the ratio $c_1/c_2 = 1$ also achieves high values of the manipulability index. With this in mind, we can conclude that both options $c_1/c_2 = 1$ and $c_1/c_2 = 2$ are viable for the mobile base algorithm, though the ratio $c_1/c_2 = 1$ might be a better starting choice as it achieves better online direction estimation performance.



(a) Drawer mechanism



(b) Door mechanism

Figure 7.10: Performance plots for different ratios c_1/c_2.

7.2 Gazebo Simulation Results

Having completed a thorough analysis of the algorithm in the PyBullet simulation, the last step before we start testing on the real Franka Emika Panda robot is to test our algorithm in the ROS Control framework. As previously mentioned, we do this by deploying the state machine described in Sections 6.1 and 6.2. Since the purpose of this simulation is to test the low level control algorithm, we perform two simple experiments. The first one requires that the end effector moves with a constant velocity of $v = 0.1\,\mathrm{m/s}$ along the x axis of the body frame. The second test is exactly the same, except this time, the end effector movement should be along the y axis of the body frame. Both experiments were successfully repeated 5 times and the averaged plots are presented in Figure 7.11.

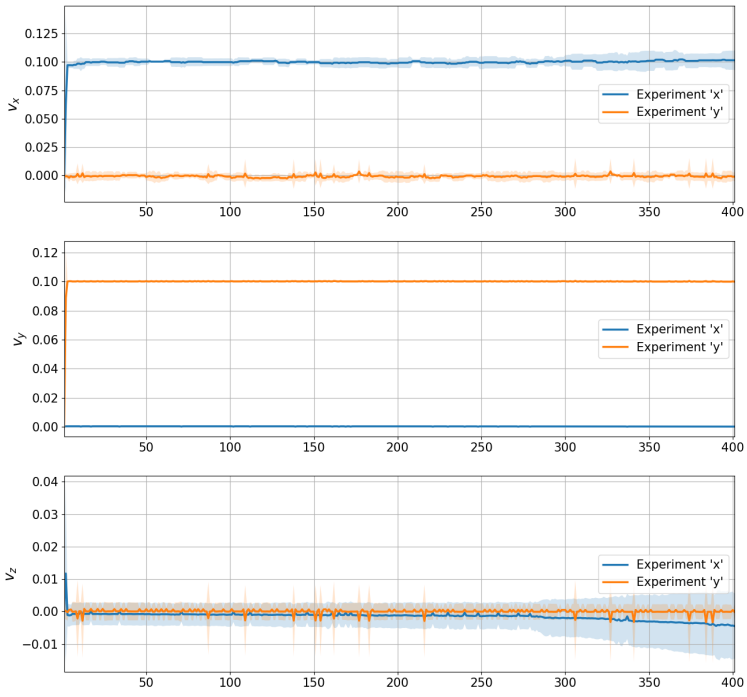


Figure 7.11: Velocity plots for the Gazebo simulation.

The most important aspect that came out of testing in the Gazebo simulation, apart from validating the real time communication pipeline, was establishing a suitable value of the parameter ΔT used for approximate differentiation and integration in Equations 5.13, 5.56 and 5.57 within the velocity planner module. When we first tried with $\Delta T = \frac{1}{240}\,\mathrm{s}$, which is equal to the value of the sampling time used in the PyBullet simulation, the end effector drifted away from the desired trajectory. By manual tuning, we established a suitable parameter value of $\Delta T = 0.05\,\mathrm{s}$ that will later be used for the real world testing. With this, we complete the testing phase in the simulated environments and are now ready to tackle the real world implementation of the algorithm.

Chapter 8

Real Robot Experiments

In this chapter, we test our algorithm on a real robot that is tasked with opening a drawer and a cabinet door model. Since we do not deploy the grasping module on the real robot, we have to manually provide the initial grasping position. In order to emulate the potential grasping alignment error, described as the discrepancy between the actual grasping pose and the ideal one which is perpendicular to the door plane, we test the algorithm for different grasping positions and random grasping orientations.

When operating a drawer mechanism, we can separate the tests into three groups corresponding to different grasping positions with respect to the door handle frame (left, middle and right grasping positions presented in Figure 8.1). Under the fixed-grasp assumption, these three grasping positions impose different sets of feasible initial arm configurations for a particular pose of the mobile base. Furthermore, starting from a grasping position closer to the edge of the door handle makes the door opening procedure more sensitive to the initial direction estimation. Smaller distance between the initial grasping point and the edge of the handle could deny the online estimation procedure a possibility to recover from poor initialization as it could lead to a faster termination of the procedure due to complete loss of the grasp caused by end effector slipping along the door handle.

(a) Left position (b) Middle position (c) Right position

Figure 8.1: Different grasping positions.

To report the average performance of the algorithm, we start with the same fixed initial unconstrained direction estimate for each of the runs given by:

$$e_{EE}^{0} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

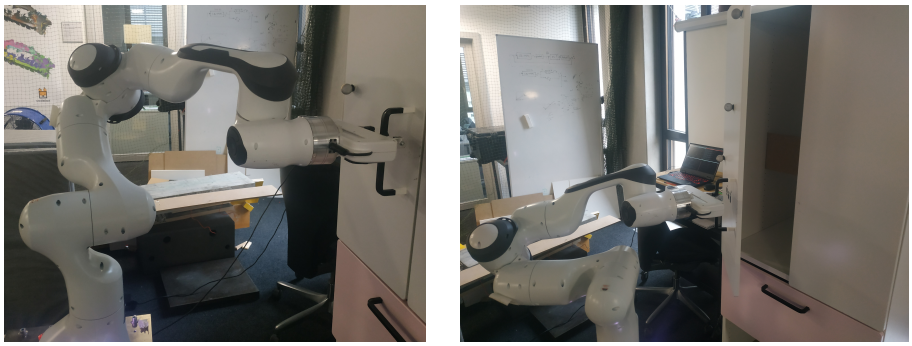
For each of the three grasping positions, we perform $N_{\text{runs}} = 10$ complete runs with both fixed and mobile base and report the same metrics used in the simulation. For

each of these metrics L, we also report its relative change when switching from the fixed base algorithm to the mobile base one. This change is given by:

$$R = \frac{L^{\text{mobile}} - L^{\text{fixed}}}{L^{\text{fixed}}} \cdot 100\%$$

In the end, we also test the complete whole body control algorithm with the initial direction estimation procedure activated.

When operating the cabinet door model, the coordinates of the initial grasping position in the horizontal plane of the world coordinate system (the plane in which the movement of the mobile base is possible) are always the same. Therefore, we test the algorithm for random arm configurations and do not explicitly separate the results for different grasping positions in the z-axis direction. Figure 8.2 depicts the cabinet door opening procedure.



(a) Closed door (b) Opened door

Figure 8.2: Cabinet door opening procedure.

We perform $N_{\text{runs}} = 10$ runs with both fixed and mobile base and report the same metrics as before. We also test the complete whole body control algorithm with the initial direction estimation procedure activated. In all of the tests that we run, we set the total number of loop iterations to $N_{\text{total}} = 500$. The hyper-parameters that describe the unconstrained direction of motion estimator are given by:

$$\alpha = 0.1; \quad m = 0.5; \quad f_c = \frac{1}{50\Delta T}$$

where $\Delta T = 0.05\,\text{s}$. The velocities during the initial period of the door opening procedure and afterwards are given by:

$$v_1 = 0.005\,\text{m/s}; \quad v_2 = 0.01\,\text{m/s};$$

The parameters of the linear velocity time profile are:

$$T_{\text{init}} = N_{\text{start-up}}; \quad t_0 = \left\lfloor \frac{T_{\text{init}}}{3} \right\rfloor; \quad \alpha_{\text{init}} = \alpha_{\text{regular}} = 0.5$$

The scaling coefficient λ described in the Equation 5.9 is determined by parameters:

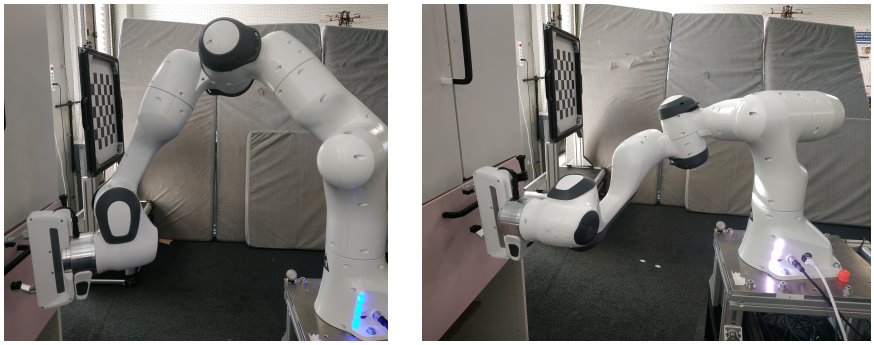
$$\alpha = 0.1; \quad R_{\text{lim}} = 0.1$$

Finally, the hyper parameters that describe the optimization procedures used in the end effector velocity planner module are given by:

$$\gamma = 0.5; \quad \beta = 0.5; \quad c_1 = 1; \quad c_2 = 1$$

8.1 Drawer Mechanism

In this section, we give the results when operating the drawer mechanism. The closed loop control algorithm is tested for two different starting configurations. This is done to demonstrate how the algorithm performs when starting from a constrained initial configuration and from a more relaxed one. Figure 8.3 approximately shows the two initial configurations. Important to note is that, since we manually provide the initial grasp for each of the runs, the initial configurations belonging to the same group are never exactly the same.



(a) Configuration 1 (b) Configuration 2

Figure 8.3: Initial grasping configurations.

In order to evaluate the performance of the initial direction estimation procedure and the unconstrained direction of motion estimator, we need to have the ground truth value of the unconstrained direction of motion defined at all times. For the drawer case, this is simple to obtain manually as the unconstrained direction of motion never changes during the run. Before each of the runs is started, we align the end effector's z axis with the true unconstrained direction of motion and record the n_{true} value.

8.1.1 Initial Unconstrained Direction of Motion

We test the initial unconstrained direction of motion procedure for 5 different missalignments θ (as shown in Figure 8.4) in the horizontal plane between the true unconstrained direction of motion and the $-$z axis of the end effector's frame. For each of them, we perform 5 estimation runs and report the average dot product between the true and the estimated value and the corresponding standard deviation. The results are presented in Table 8.1.

Table 8.1: Initial direction estimation performance.

Angle	run 1	run 2	run 3	run 4	run 5	μ	σ
$\approx 0°$	0.9834	0.9916	0.9939	0.9891	0.9899	0.9896	0.0035
$\approx -45°$	0.8816	0.7611	0.8343	0.7452	0.8322	0.8109	0.0506
$\approx 45°$	0.8614	0.9012	0.8269	0.8892	0.9129	0.8783	0.0308
$-45° < \theta < 0°$	0.8951	0.9010	0.8758	0.8951	0.8812	0.8896	0.0095
$0° < \theta < 45°$	0.9149	0.9322	0.9314	0.9211	0.9264	0.9252	0.0065

For each of the runs, we set the hyper-parameters of the initial direction estimation

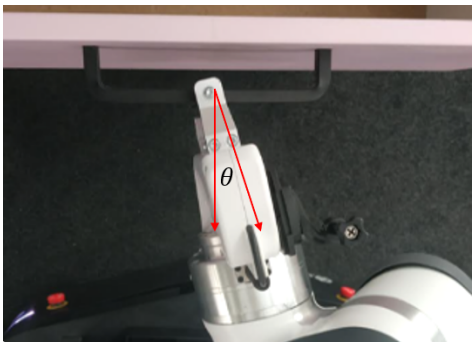


Figure 8.4: Grasping missalignment θ.

procedure to:

$$N_{\mathrm{m}} = 3 \quad \text{and} \quad V = 0.005\,\mathrm{m/s}$$

The average duration of the initial unconstrained direction of motion estimation procedure is 1.91 s. For the starting conditions that are close to the actual unconstrained direction of motion, the estimator achieves an average dot product of approximately 0.9896 with a small standard deviation. It is interesting to note that when the absolute value of the angular displacement increases, the average dot product decreases, which means that we obtain a more imprecise estimate. Furthermore, the uncertainty about the obtained average estimate also increases since the standard deviation increases.

8.1.2 Left Grasping Pose

Figures 8.5(a) and 8.5(b) respectively show the performance of the system when the base is fixed and when the base is mobile. Furthermore, Tables 8.2 and 8.3 respectively report the metric values for the first and the second initial configuration of the robotic arm. Here, the first configuration has higher initial manipulability index and hence corresponds to the more relaxed starting pose.

Table 8.2: Metrics for the drawer model with left initial grasp and first initial configuration.

Metric	Fixed base	Mobile base	Change [%]
$\mathrm{RMSE}_{\mathrm{init}}$	**0.0181**	0.0305	+68.51
$\langle n_{\mathrm{true}}, n_{\mathrm{est}} \rangle^{\mathrm{avr}}$	**0.9944**	0.9919	-0.25
μ^{avr}	0.0698	**0.0877**	+25.64
μ^{end}	0.0476	**0.0889**	+86.76
$T^{\mathrm{avr}}_{\mathrm{total}}$	**18.6112**	19.9883	+7.26
Δt^{avr}	**0.0372**	0.0399	+7.26

Taking a closer look at the average dot product of the true and the estimated unconstrained direction of motion over the whole run, we can see that the system benefits from using the additional DOFs provided by the mobile base when having a more constrained initial configuration and experiences a slight decrease in performance in the first scenario. However, since the relative change is very small ($< 1\%$), one can consider it to be negligible and cannot attribute it to a particular cause with certainty. The RMSE during the initial period is higher when using the mobile base if the robot starts from a more relaxed configuration. In addition, tables show that

Table 8.3: Metrics for the drawer model with left initial grasp and second initial configuration.

Metric	Fixed base	Mobile base	Change [%]
$\text{RMSE}_{\text{init}}$	0.0727	**0.0188**	-74.14
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9851	**0.9941**	+0.91
μ^{avr}	**0.0563**	0.0552	-1.95
μ^{end}	0.0489	**0.0573**	+17.17
$T^{\text{avr}}_{\text{total}}$	**18.6196**	22.2160	+19.31
Δt^{avr}	**0.0372**	0.0444	+19.31

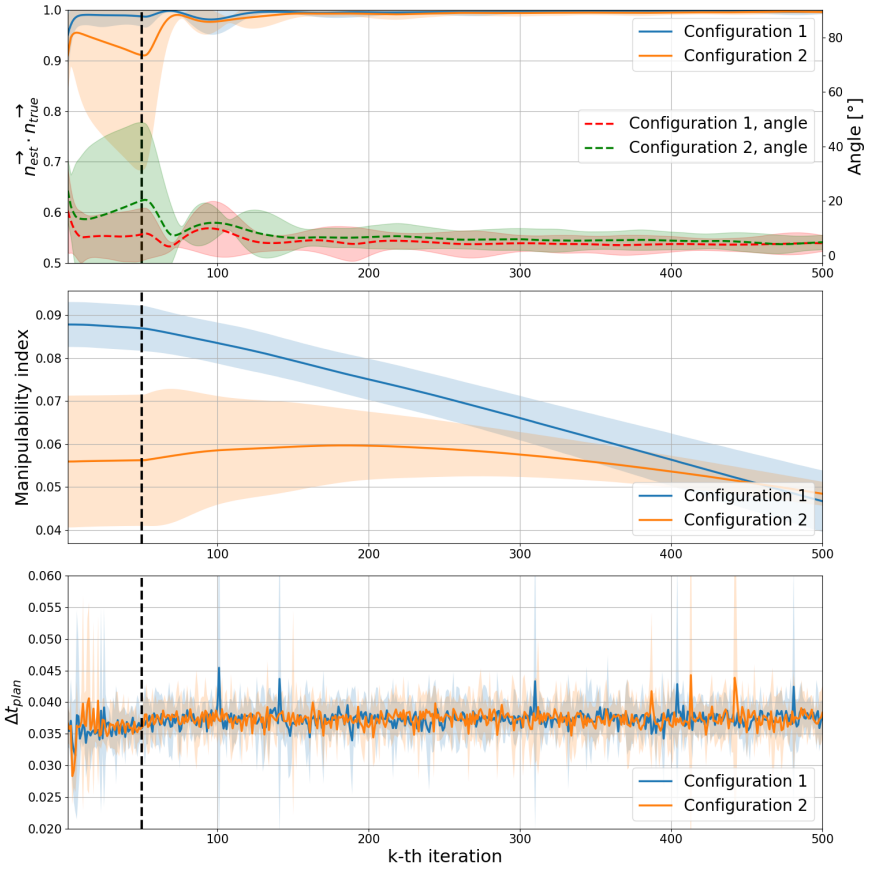
on average, the unconstrained direction of motion estimator achieves better convergence when deployed with the active mobile base. Interesting to note is that in all four cases, the average angular discrepancy between the true and the estimated unconstrained direction is less than 10° after the first 150 iterations of the loop. More importantly, the $\pm 2\sigma$ interval of the measured data is upper bounded by the value of 10° after 250 iterations for both algorithms.

The average manipulability index at the end of the run is higher when using the mobile base for both initial configurations. The improvement is significant in the first scenario with a relative increase in performance of more than 86%. When starting from the second initial configuration, the increase is still relatively big (approx. 17%) but not as drastic as in the previous case. For the more constrained initial configuration, the reported average manipulability over the whole run is lower when using the mobile base. However, if we take a closer look at the two figures, we can see that the initial manipulability index for the fixed base method in combination with the second configuration is higher than the initial manipulability index for the mobile base and the same configuration. Let us denote with $\mu_{\max}$ the maximal manipulability index over the whole run and with $\mu(k=0)$ the manipulability index at the beginning of the run. By comparing the difference between the maximal achieved manipulability index over the whole run and the manipulability index at the start of the run for the two cases:

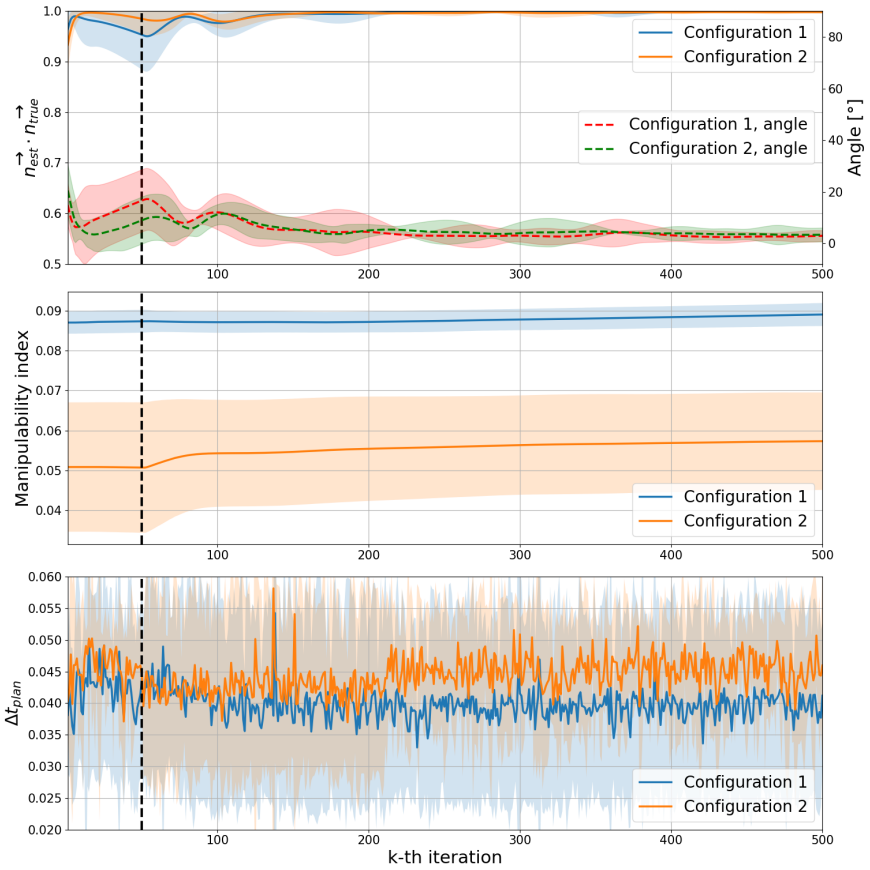
$$\frac{\mu^{\text{fixed}}_{\max} - \mu^{\text{fixed}}(k=0)}{\mu^{\text{mobile}}_{\max} - \mu^{\text{mobile}}(k=0)} \approx 0.43$$

we can conclude that the individual contribution of the mobile base algorithm to the overall increase of the manipulability (relative to the initial configuration) is actually higher. The observed decrease in the average manipulability index is also deemed negligible ($< 2\%$) and can be considered a consequence of a smaller initial manipulability index in the case of a mobile base and the second configuration.

The average planning time per iteration is longer for the mobile base configuration. This is to be expected as the mobile base algorithm has to perform one additional optimization procedure. For the fixed base case, the total planning time is approximately the same for both initial configurations. Figure 8.5(b) shows that for the mobile base algorithm, the average planning time per iteration is higher on average when the robot is dealing with a more constrained initial configuration. This comes as no surprise since we are using iterative optimization solvers that could simply require more optimization iterations in order to find the optimal solution (in all of the cases, the maximal number of optimization steps is the same) when dealing with a more constrained robot configuration.



(a) Fixed base



(b) Mobile base

Figure 8.5: Metrics when operating the drawer mechanism with left initial grasp.

8.1.3 Middle Grasping Pose

Similarly to the previous grasping position, Figures 8.6(a) and 8.6(b) show the performance of the system when the base is fixed and when the base is mobile. The corresponding tables with reported values of the metrics for the first and the second initial configuration of the robotic arm are 8.4 and 8.5 respectively. Once again, the first configuration has higher initial manipulability index and hence corresponds to the more relaxed starting pose. Note that the average starting manipulability index for the first configuration is smaller compared to the left grasping case when a fixed base algorithm is used. When a mobile base algorithm is used, this difference is even bigger (approx. 0.02). For the fixed base algorithm combined with the second scenario, the starting manipulability index is roughly the same as in the left grasping case. When the mobile base is activate in the second scenario, the average manipulability index at the beginning of the procedure is slightly higher than in the left grasping case. These discrepancies are the consequence of the manually adjusted initial grasp and should be noted when analysing the reported results.

Table 8.4: Metrics for the drawer model with middle initial grasp and first initial configuration.

Metric	Fixed base	Mobile base	Change [%]
$\text{RMSE}_{\text{init}}$	0.0685	**0.0203**	-70.36
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9889	**0.9942**	+0.53
μ^{avr}	0.0665	**0.0726**	+9.17
μ^{end}	0.0444	**0.0750**	+59.10
$T_{\text{total}}^{\text{avr}}$	**18.6682**	23.2044	+24.39
Δt^{avr}	**0.0373**	0.0464	+24.39

Table 8.5: Metrics for the drawer model with middle initial grasp and second initial configuration.

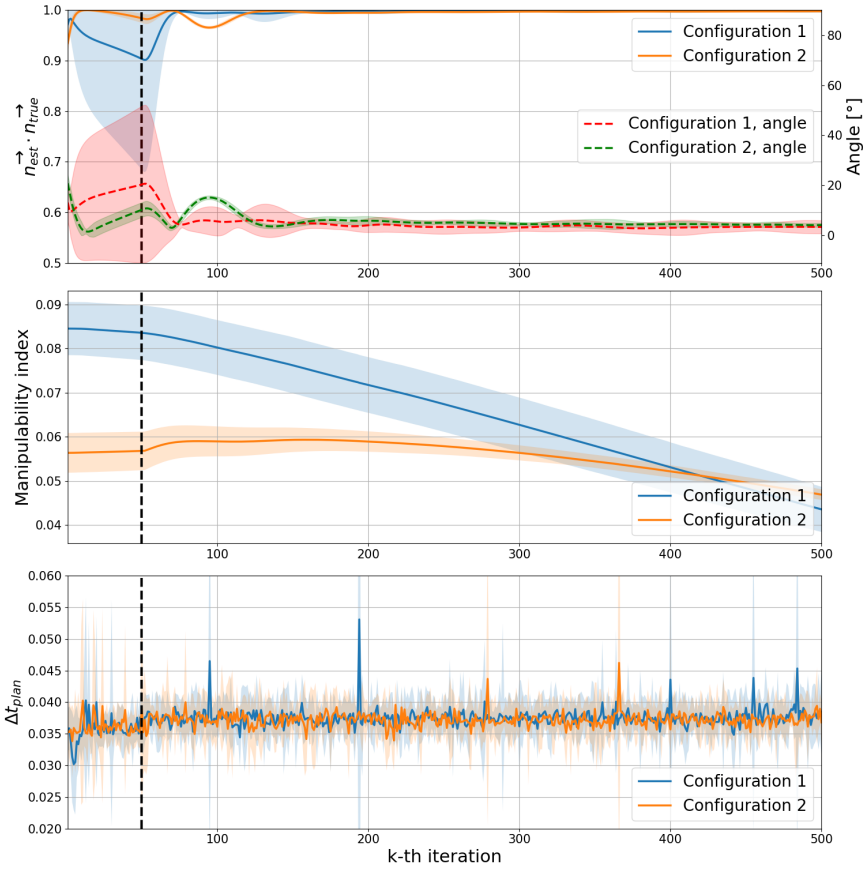
Metric	Fixed base	Mobile base	Change [%]
$\text{RMSE}_{\text{init}}$	**0.0155**	0.0513	+231
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	**0.9939**	0.9876	-0.63
μ^{avr}	0.0556	**0.0586**	+5.39
μ^{end}	0.0473	**0.0616**	+30.23
$T_{\text{total}}^{\text{avr}}$	**18.5551**	22.2845	+20.09
Δt^{avr}	**0.0371**	0.0446	+20.09

In the middle grasping case, the average dot product of the true and the estimated unconstrained direction of motion over the whole run experiences a small increase (approx. 0.53%) when starting from the first initial configuration. When the robot is starting from a more constrained configuration, activating the mobile base seems to be decreasing the average dot product with a relative change of approximately 0.63%. Since we, once again, have very small values of the relative change, the difference is deemed negligible. The results obtained for the RMSE during the initial period are deemed indecisive. A significant relative decrease of 70.36% is experienced in the first scenario when introducing a mobile base. In the case of a more constrained initial configuration, the significant relative change goes in favour of the fixed base algorithm, though the absolute differences in these two cases are close in terms of value. The average angular discrepancy between the true and

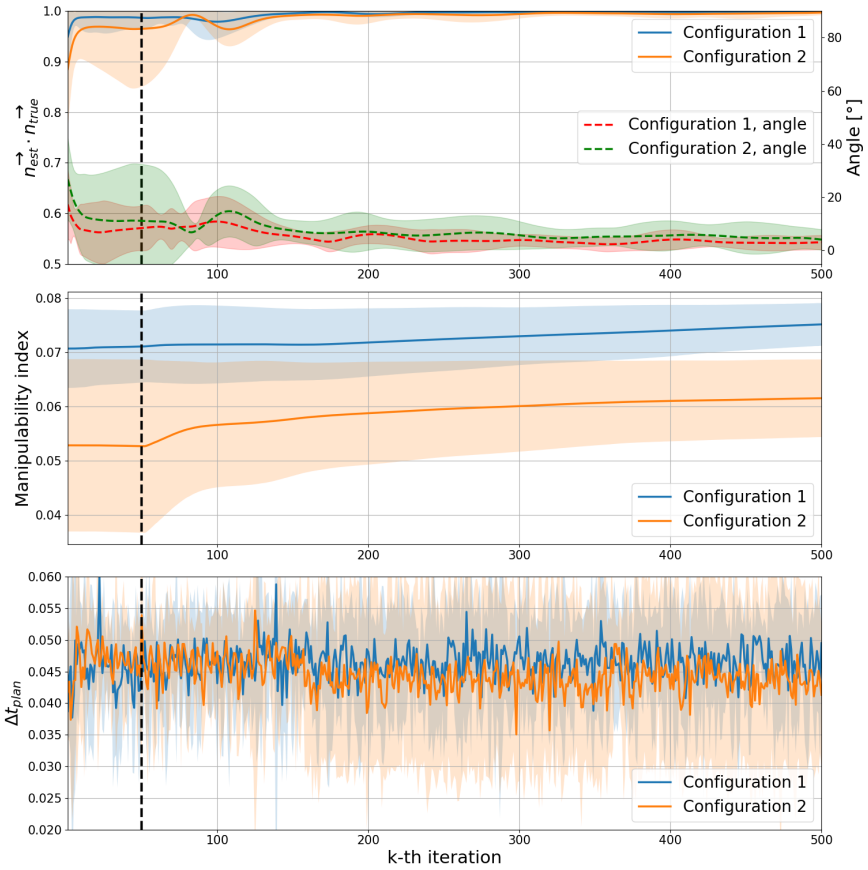
the estimated unconstrained direction of motion is, once again, less than 10° after approximately 150 iterations for each of the four cases. When the base is fixed, the $\pm 2\sigma$ interval for both configurations is very narrow after 150 iterations and is upper bounded by 10°. When the mobile base is activated, the $\pm 2\sigma$ interval is wider when starting from the more constrained configuration and can be upper bounded by 15°.

When it comes to the average manipulability index of the robot over the whole run, an improvement is observed for both initial configurations. In the first scenario, the relative change is approximately 9.17% whereas in the second scenario we experience the increase of approximately 5.39%. Though these numbers do not represent a drastic change in performance, they do go in favor of utilizing the mobile base when possible. When it comes to the average manipulability index at the end of the run, once again we experience a significant relative increase. Compared to the left grasping case, the relative increase of the manipulability index at the end in the first scenario is smaller (59.10%). However, the relative increase of the manipulability index in the second scenario is bigger and is now approximately equal to 30.23%.

Similar to the previous case, the average total planning time is longer when the mobile base is used regardless of the initial configuration of the robotic arm. What is different compared to the left grasping case is the average total planning time when starting from the more relaxed initial configuration and utilizing the mobile base as well. In the left grasping case, the relative increase in the total planning time for the first scenario was approximately 7.26%. When the handle is grasped in the middle, on average, a relative increase of 24.39% is experienced in the first scenario. In the second scenario, the relative increase in the planning time is approximately 20.09% which is considered close to the 19.31% reported for the left grasping case. The total planning time for the fixed base algorithm is roughly the same for both initial configurations of the robot which can now also be said for the mobile base algorithm. It is interesting to note that although the first configuration corresponds to a more relaxed scenario as the manipulability index at the beginning of the door opening procedure is higher, the average planning time per iteration is longer. We can see that the average planning time per iteration in the second scenario is 3.87% shorter than in the first scenario.



(a) Fixed base



(b) Mobile base

Figure 8.6: Metrics when operating the drawer mechanism with middle initial grasp.

8.1.4 Right Grasping Position

This is the final grasping position that we test. Similar to the previous two cases, we show the performance plots for the fixed base and the mobile base algorithms in Figures 8.7(a) and 8.7(b) respectively. In addition, the corresponding metric values are provided in Tables 8.6 and 8.7. The average starting manipulability index for the fixed base algorithm is slightly smaller than in both previous cases for the more relaxed initial configuration. In the second scenario, the average starting manipulability index is close in value to the ones introduced in the left and the middle grasping cases. For the mobile base algorithm, the starting manipulability index in the first scenario is close to the one introduced in the middle grasping case. However, in the second scenario, the starting manipulablity index is just under 0.05, which makes it smaller than in any of the two previous grasping configurations.

Table 8.6: Metrics for the drawer model with right initial grasp and first initial configuration.

Metric	Fixed base	Mobile base	Change [%]
$\text{RMSE}_{\text{init}}$	**0.0266**	0.0419	+57.52
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	**0.9943**	0.9919	-0.24
μ^{avr}	0.0652	**0.0698**	+7.06
μ^{end}	0.0455	**0.0707**	+55.38
$T_{\text{total}}^{\text{avr}}$	**18.5279**	23.2002	+25.22
Δt^{avr}	**0.0370**	0.0464	+25.22

Table 8.7: Metrics for the drawer model with right initial grasp and second initial configuration.

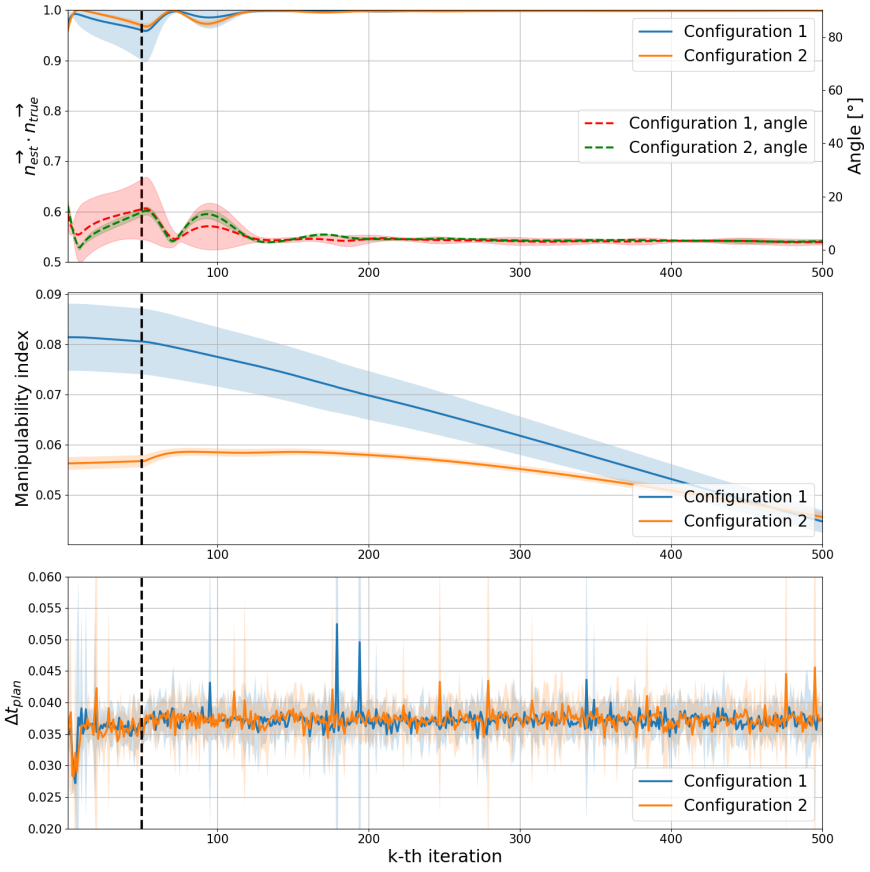
Metric	Fixed base	Mobile base	Change [%]
$\text{RMSE}_{\text{init}}$	**0.0169**	0.0693	+310
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	**0.9946**	0.9818	-1.28
μ^{avr}	**0.0548**	**0.0548**	0
μ^{end}	0.0462	**0.0579**	+25.32
$T_{\text{total}}^{\text{avr}}$	**18.5935**	22.9430	+23.39
Δt^{avr}	**0.0372**	0.04588	+23.39

In the right grasping case, the average dot product between the true and the estimated unconstrained direction of motion experiences a small relative decrease for both initial configurations when the mobile base is active. Similar to the two previous cases, the values of these relative decreases are deemed negligible. On the other hand, the average RMSE during the initial period of the door opening procedure seems to be increasing when introducing the mobile base in both scenarios. Interesting to note is that the average RMSE values reported for the mobile base algorithm are the largest when the initial grasping position is on the right hand side of the handle. For the fixed base algorithm, the average angular displacement between the true and the estimated unconstrained direction of motion is approximately the same (less than 5°) for both initial configurations after 125 loop iterations. Furthermore, the $\pm 2\sigma$ interval of the average angular displacement can be upper bounded by 5°. In the mobile base case, the average angular displacement for the more constrained initial configuration is constantly bigger than the one for the more relaxed initial configuration. The $\pm 2\sigma$ interval for the first scenario can be upper bounded by

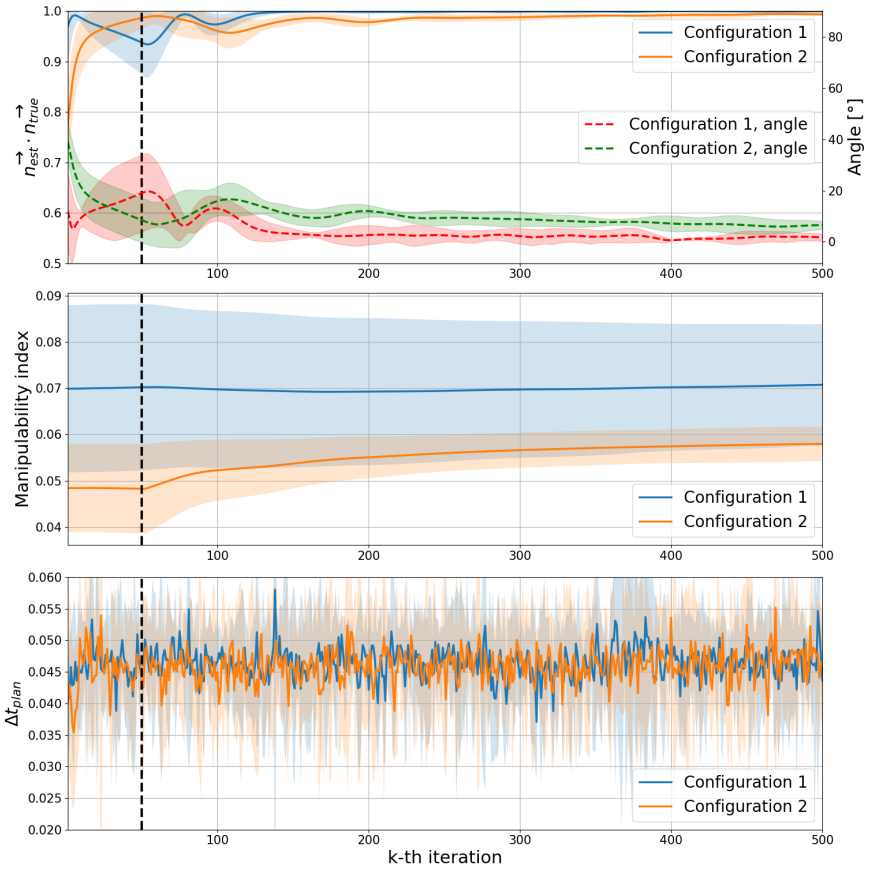
10° after approximately 125 loop iterations whereas the $\pm 2\sigma$ interval for the second initial configuration can be upper bounded by 15° after approximately 150 loop iterations.

The relative change in the average manipulability index when the mobile base is activated is positive for the more relaxed initial configuration whereas in the second scenario, this difference is approximately zero. The average manipulability index at the end of the door opening procedure once again goes in favour of the mobile base algorithm for both initial configurations. In the more relaxed scenario, a significant relative change in the average manipulability index at the end of approximately 55.38% is observed whereas in the second scenario, this relative increase is smaller (approx. 25.32%) but still significant. Compared to the middle grasping case, the experienced relative increases in performance are close in terms of value.

Just like in the previous two grasping cases, the average total planning time is longer when the mobile base is used regardless of the initial configuration. Similar to the middle grasping case, the average total planning time for the mobile base algorithm is slightly longer when the robot is starting from the more relaxed initial configuration. However, this absolute difference of approximately 0.25 s can once again be considered negligible. What is also similar to the previous grasping case is the relative change of approximately $20\% - 25\%$ in the average planning time per iteration for both initial configurations. In fact, only the left grasping scenario combined with the more relaxed initial configuration achieves lower relative increase in the average planning time per iteration (approx. 7.26%). In all other cases, this change falls within the interval $19\% - 25\%$. Just like in the middle grasping case, there is no significant discrepancy between the average planning time when starting from either of the two initial configurations. This could be due to the fact that in the left grasping case, the absolute difference between the starting manipulability indexes is approximately 0.035 whereas for the other two grasping positions this difference is smaller than 0.025. With that in mind, we can conclude that the two initial configurations were more similar in terms of the robot's manipulability when the handle was grasped in the middle and on the right.



(a) Fixed base



(b) Mobile base

Figure 8.7: Metrics when operating the drawer mechanism with right initial grasp.

8.1.5 Complete Algorithm With the Initial Direction Estimation Procedure

In the previous three subsections we presented a successful implementation of our algorithm on a real robot. To make the testing conditions more uniform across different runs, the presented results were given for the same initial estimate of the unconstrained direction of motion. To complete our analysis, in this subsection we show how the mobile base algorithm performs when the initial direction estimation procedure is also performed. The complete run is performed 5 times with random grasping positions and grasping orientations and the average performance of the system is reported. Figure 8.8 represents the performance plots of the system over the whole run. The corresponding metrics are given in the Table 8.8. Trends in the three performance plots that were present in the previous analysis match the ones present in the Figure 8.8.

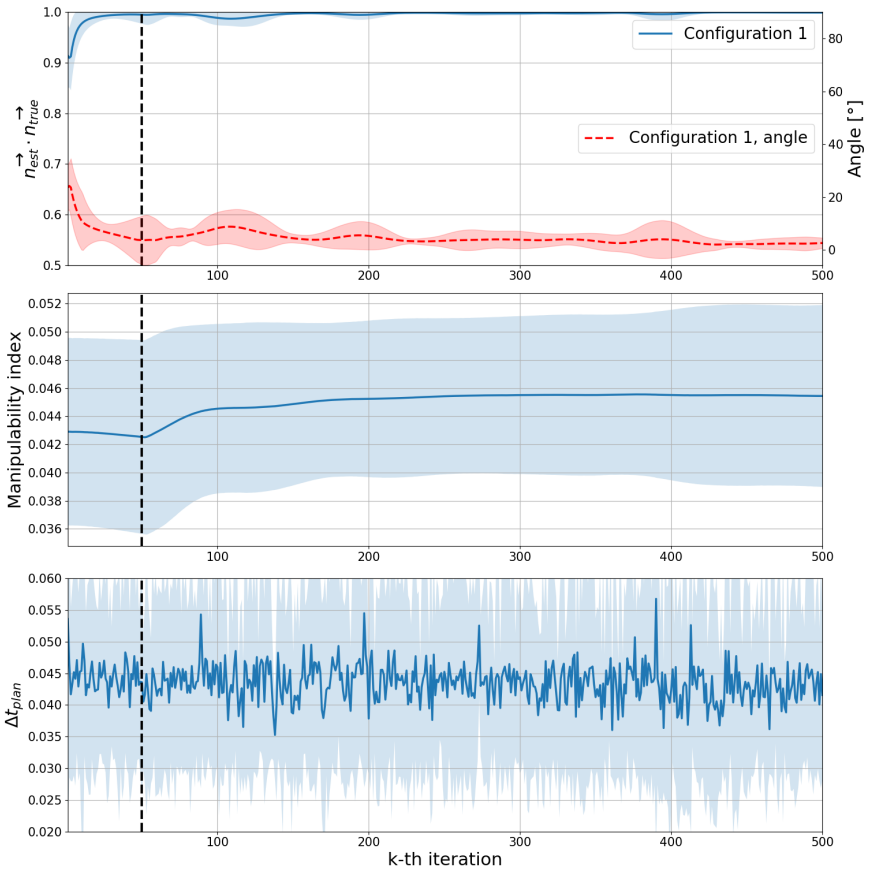


Figure 8.8: Metrics when operating the drawer mechanism with complete procedure.

Table 8.8: Metrics for the drawer model when using the complete algorithm.

$\text{RMSE}_{\text{init}}$	$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	μ^{avr}	μ^{end}	$T^{\text{avr}}_{\text{total}}$	Δt^{avr}
0.0297	0.9951	0.0449	0.0454	21.7804	0.0435

On average, the manipulability index of the robotic arm tends to increase during

the run, achieving the average manipulability index at the end of approximately 0.0454. Though this value is smaller than the ones reported in Sections 8.1.2, 8.1.3 and 8.1.4, it is important to note that the manipulability index at the start of the door opening procedure is also smaller (approx. 0.0425). The average dot product between the true and the estimated unconstrained direction of motion is high, with an approximate value of 0.9951. Figure 8.9 shows a plot for each velocity component over the whole run.

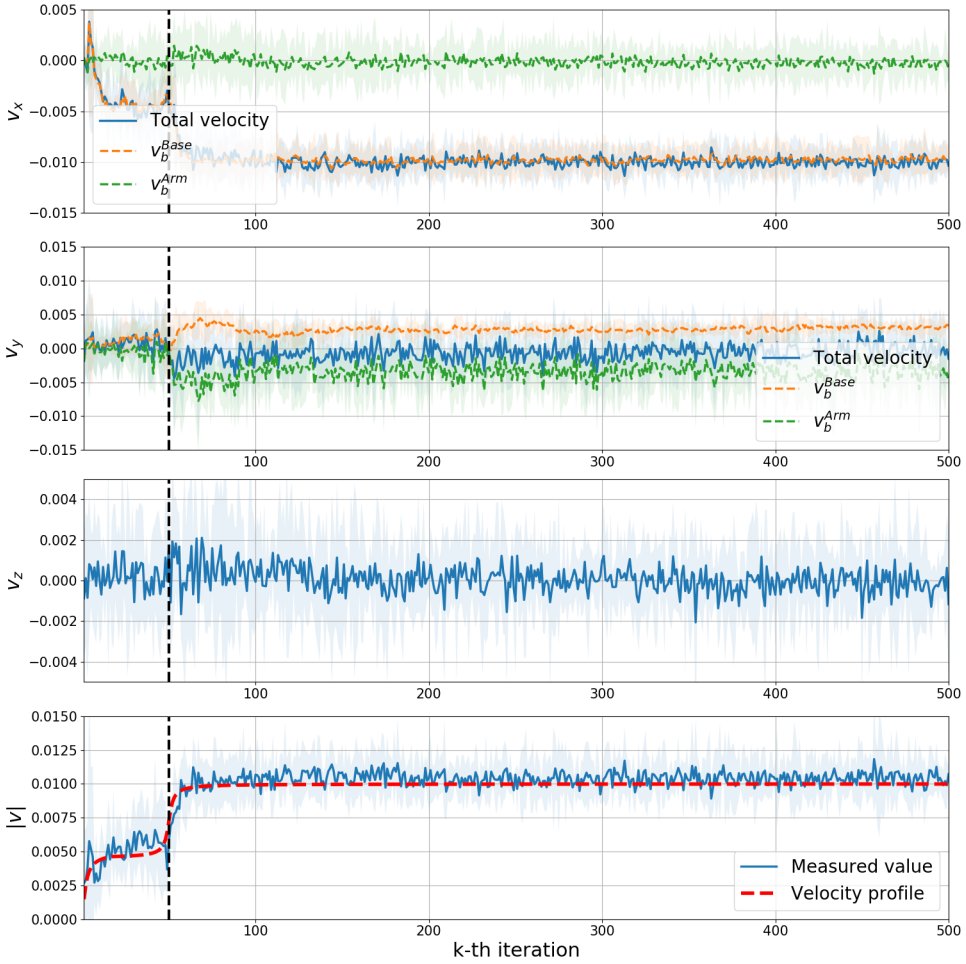


Figure 8.9: End effector Cartesian velocity.

As can be seen in the picture, the x component of the relative velocity between the arm and the base is on average close to zero and the movement is primarily achieved through the movement of the mobile base. In the y direction on the other hand, both components approximately contribute equally, but in the opposite directions, causing negligible movement along this axis. Finally, by taking a look at the magnitude of the linear velocity, we can see that it complies with the temporal profile defined by the velocity planner module.

8.1.6 Summary of the Drawer Operation Procedure

So far, we have shown that our proposed solution with the mobile base successfully opens the drawer without any a priori information about the mechanism that is being operated. Moreover, the solution has proven robust to different grasping

positions and grasping orientations. Compared to the same algorithm but without the mobile base activated, this solution always achieves better average manipulability index at the end of the procedure leaving the robotic arm in a better initial configuration for the task that follows.

8.2 Cabinet Door Mechanism

This section is devoted to analysing the performance of our closed loop algorithm when operating a rotational joint mechanism such as the cabinet door. As previously mentioned, we do not separately test the algorithm for different grasping positions but rather report the joint average results for various grasping positions and orientation. Furthermore, the mounting position of the handle and the door's height made it impossible to find two configurations that significantly differ in the starting manipulability index so it made sense to merge all the measurements into one starting initial configuration as opposed to the two introduced for the drawer mechanism. The hyper-parameters used in these experiments are the same as in the drawer case since we are trying to define a unique algorithm capable of solving both door opening problems. The same goes for the fixed initial unconstrained direction of motion estimate.

Obtaining the ground truth for the unconstrained direction of motion is more challenging than in the case of the drawer mechanism as the unconstrained direction of motion vector is constantly changing. Since we are dealing with a fixed-grasp manipulation task, we first record the end effector movement in the world frame over the whole run and then afterwards, fit the best circle in the plane parallel to the ground along which the end effector was moving. If we denote with (x_c, y_c) the center and with R the radius of the optimal circle, than for every recorded end effector position (x_i, y_i) we can write the equation of the optimal circle as:

$$\underbrace{\begin{bmatrix} 1 & -2x_i & -2y_i \end{bmatrix}}_{a_i} \underbrace{\begin{bmatrix} x_c^2 + y_c^2 - R^2 \\ x_c \\ y_c \end{bmatrix}}_{\phi} = \underbrace{-x_i^2 - y_i^2}_{b_i} \tag{8.1}$$

By stacking the individual matrices a_i and b_i into $\tilde{A}$ and $\tilde{b}$ respectively, we can write the problem of finding the optimal circle as:

$$\tilde{A}\phi = \tilde{b} \tag{8.2}$$

whose solution is given by:

$$\phi = \left(\tilde{A}^T \tilde{A}\right)^{-1} \tilde{A}^T \tilde{b} \tag{8.3}$$

Now that we have the parameters of the optimal circle, for each end effector position we define the true unconstrained direction of motion vector (expressed in the world frame) as:

$$n_{\text{true}} = \begin{bmatrix} \cos(\theta_i + \pi/2) \\ \sin(\theta_i + \pi/2) \\ 0 \end{bmatrix} \tag{8.4}$$

where the angle θ_i is given by:

$$\theta_i = \arctan2\left(\frac{y_i - y_c}{x_i - x_c}\right) \tag{8.5}$$

8.2.1 Initial Unconstrained Direction of Motion

Now that we have defined the true unconstrained direction of motion we can proceed to test the initial unconstrained direction of motion estimation procedure. The mounting position of the handle limits the angular misalignment in the vertical plane where the handle is located so we are no longer able to test the procedure for larger absolute differences. The results are presented in Table 8.9, where $\delta > 0$ is used

Table 8.9: Initial direction estimation performance.

Angle	run 1	run 2	run 3	run 4	run 5	μ	σ
$\approx 0°$	0.8677	0.8731	0.8585	0.8692	0.8621	0.8661	0.0052
$-\delta < \theta < 0°$	0.8655	0.8233	0.8724	0.8023	0.8681	0.8463	0.0282
$0° < \theta < \delta$	0.8614	0.8688	0.8619	0.8424	0.8266	0.8522	0.0155

to indicate the sign of the absolute error. The results were obtained by performing 5 runs for each of the grasping orientations with the same hyper-parameters used in the drawer opening procedure. The average duration of the procedure stays approximately the same and is equal to 1.92 s. Note that in this case, the values of the average dot products between the true and estimated initial unconstrained direction of motion are smaller than the values reported for the drawer opening procedure. Similar to the previous case, when the absolute value of the angular displacement increases, the average dot product decreases meaning that we have a more imprecise estimate. In addition, since the standard deviation also increases, so does the uncertainty about the obtained estimate.

8.2.2 Cabinet Door Opening Procedure

Just like we did for the drawer mechanism, we show the performance of the system when the base is fixed and when the base is mobile in Figures 8.10(a) and 8.10(b) respectively. The corresponding metric values are given in Table 8.10. The manipulability index at the start of the door opening procedure is in all cases, on average, close to 0.045, which is smaller then the ones presented in the drawer opening procedure.

Table 8.10: Metrics for the cabinet door model.

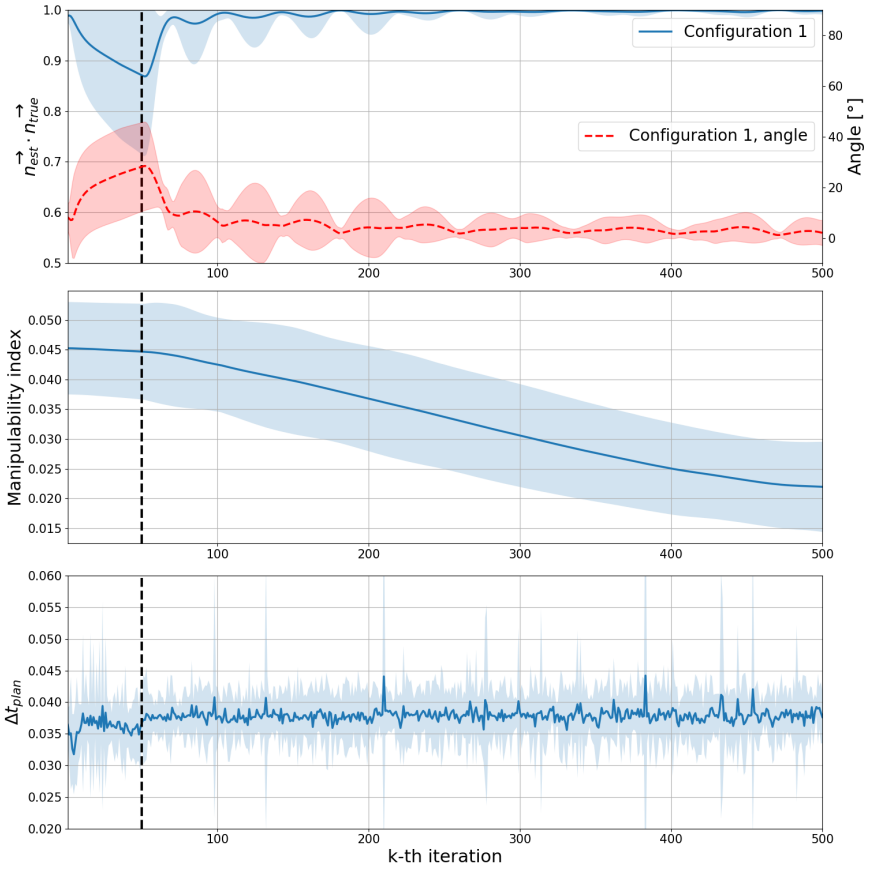
Metric	Fixed base	Mobile base	Change [%]
$\text{RMSE}_{\text{init}}$	0.0898	**0.0856**	-4.67
$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	0.9844	**0.9847**	+0.03
μ^{avr}	0.0337	**0.0479**	+42.13
μ^{end}	0.0221	**0.0525**	+137
$T^{\text{avr}}_{\text{total}}$	**18.8534**	22.0268	+16.83
Δt^{avr}	**0.0377**	0.0441	+16.83

At first glance, we can observe that the mobile base system performs better according to all criteria. The average dot product between the true and the estimated unconstrained direction of motion is approximately the same in both cases since the relative change in performance is very small (0.03%). Due to the specific initial grasping configuration imposed by the mounting position of the handle and the door's height, the robot always starts with an initial guess for the unconstrained direction of motion that is close to the true value. It is interesting to note that we

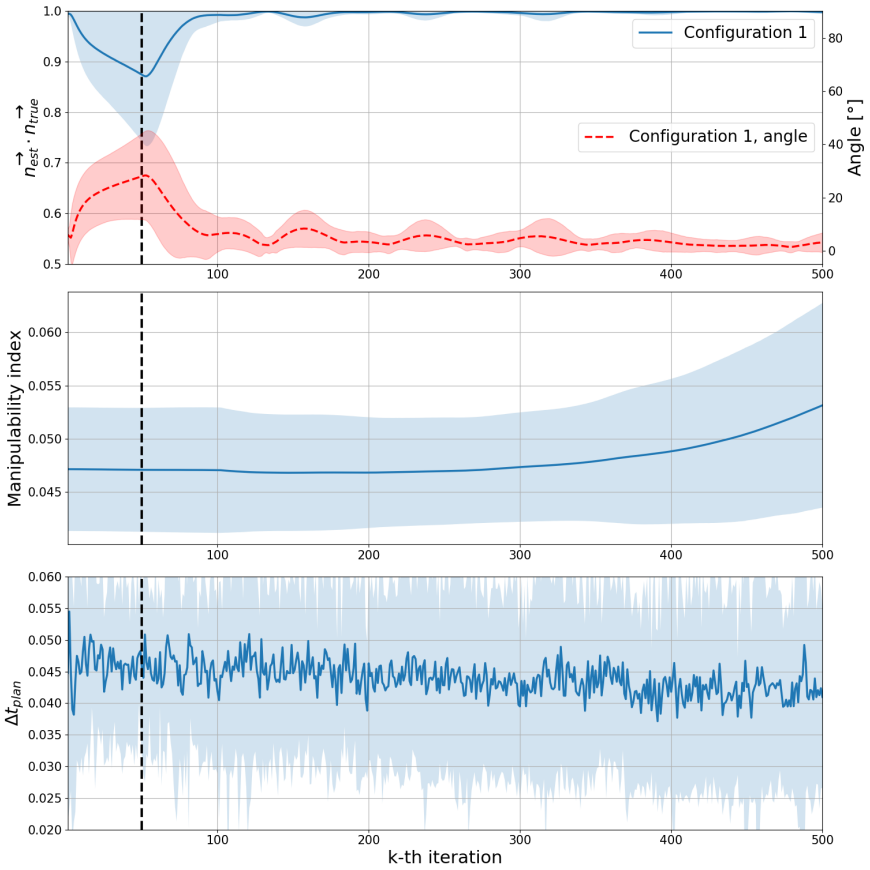
can observe a decreasing trend in the average dot product during the initial period of the door opening procedure both for the fixed base and for the mobile base algorithms. This could happen due to the fact that the cross section of the handle is not circular so the contact point switching could cause reaction forces in misleading directions. Furthermore, by design, the fixed-grasp unconstrained direction of motion estimator is not active during the initial period of the door opening procedure so there is effectively nothing to compensate for the poor estimate obtained through the force feedback. Second reason could be the initial damping present in the joint mechanisms of the furniture that we used for testing. All things considered, this is where a good initial unconstrained direction estimate helps the online update procedure as it allows the fixed-grasp estimate to quickly steer the combined one in the right direction after the initial period is over. Though the average RMSE value during the initial period is smaller when the mobile base is utilized, in this context, it provides no meaningful information about the convergence of the direction estimation algorithm. By looking at the $\pm 2\sigma$ interval of the average dot product when the base is fixed, we can observe its more oscillatory character compared to the case when the mobile base is active. The average angular displacement between the true and the estimated unconstrained direction of motion is similar in both cases after 100 loop iterations and can be bounded by 10°. When the base is fixed, the $\pm 2\sigma$ interval of the average angular displacement can be upper bounded by 15° after approximately 150 loop iterations and by 10° after 250 loop iterations. On the other hand, though the $\pm 2\sigma$ interval of the average angular displacement is in general more narrow when the mobile base is active, there are three peaks in its value that only allow us to define its upper bound of 15°, after approximately 100 loop iterations.

The overall manipulability of the robot is significantly increased with the mobile base algorithm. The average manipulability index over the whole run experiences a significant relative increase of 42.13%. Due to the nature of the door opening motion, in the fixed base case, the robot inevitably has to approach its joint limits which drastically reduces its manipulability. Even greater relative increase can be observed for the average manipulability index at the end of the run (approx. 137%). Note that the absolute values are in general smaller than the ones reported in the Sections 8.1.2, 8.1.3 and 8.1.4. However, this can be completely attributed to a more constrained initial configuration of the robot.

Similar to all the previously analyzed cases, the average total planning time when using the mobile base algorithm is around 3 s longer than for the fixed base one. In this particular case, this increase in the planning time corresponds to a relative change of approximately 16.83% and is similar to the numbers reported when operating the drawer mechanism. Although the overall manipulability of the arm is lower during the runs performed with the cabinet door mechanism, the average planning time per iteration is shorter than in all the drawer cases except the one presented in the Section 8.1.2 when the robot starts from a more relaxed initial configuration.



(a) Fixed base



(b) Mobile base

Figure 8.10: Metrics when operating the cabinet door mechanism.

8.2.3 Complete Algorithm With the Initial Direction Estimation Procedure

The previous subsection presented a successful implementation of our algorithm on the real robot that relies on a predefined initial unconstrained direction of motion. To complete our analysis, in this subsection we present the results for the mobile base version of our algorithm with enabled initial unconstrained direction of motion procedure. Just like we did for the drawer mechanism, the complete door opening procedure is repeated 5 times with random initial grasping positions and grasping orientations and the average performance of the closed loop system is reported. Figure 8.11 represents the performance plots of the system over the whole run. The corresponding metrics are given in the Table 8.11. We can observe the same temporal trends in the three performance plots that were also present in the previous analysis of the cabinet door opening procedure.

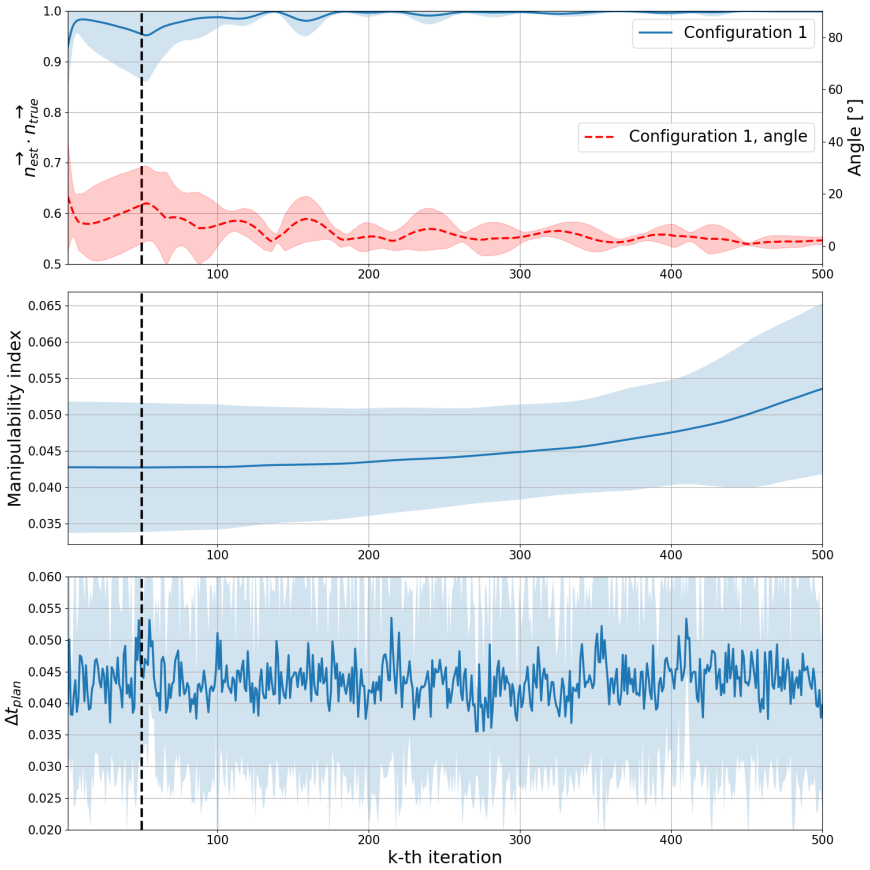


Figure 8.11: Metrics when operating the cabinet door with complete procedure.

Table 8.11: Metrics for the cabinet door model when using the complete algorithm.

$\text{RMSE}_{\text{init}}$	$\langle n_{\text{true}}, n_{\text{est}} \rangle^{\text{avr}}$	μ^{avr}	μ^{end}	$T^{\text{avr}}_{\text{total}}$	Δt^{avr}
0.02371	0.9912	0.0452	0.0529	21.7435	0.0435

Once again, on average, the manipulability index of the robotic arm increases during the run, achieving the average manipulability index at the end of approximately

0.0529. The average dot product between the true and the estimated unconstrained direction of motion is high after the initial period, with an approximate value of 0.961. Figure 8.12 shows a plot for each velocity component over the whole run.

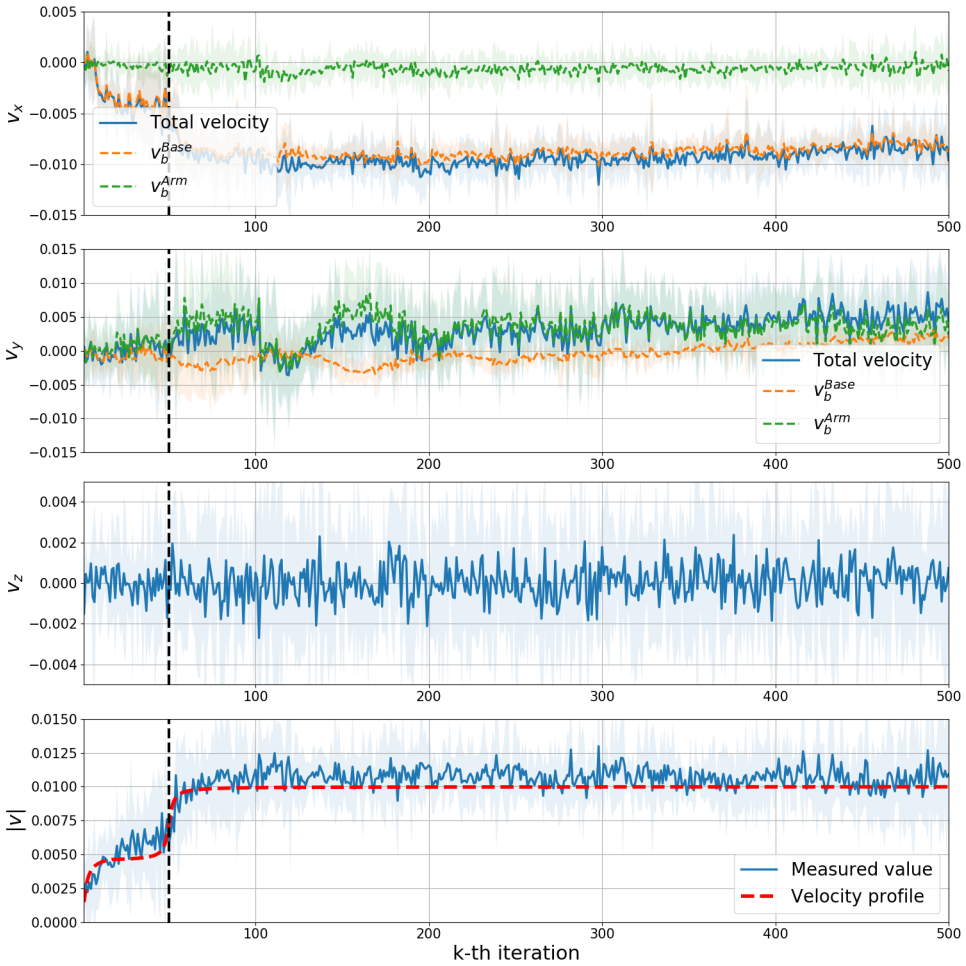


Figure 8.12: End effector Cartesian velocity.

Similar to the drawer mechanism case, the movement along the x axis of the base frame is almost solely achieved through the movement of the mobile base. On the other hand, the movement along the y direction of the base frame is primarily achieved through the corresponding component of the relative velocity between the arm and the base. Finally, by taking a look at the magnitude of the linear velocity, we observe that it complies with the previously defined temporal profile.

8.2.4 Summary of the Cabinet Door Opening Procedure

In the end, we have also demonstrated how our mobile base solution successfully opens the cabinet door without any given information about the mechanism that is being operated. The solution has proven to be robust to different grasping positions and orientation within the feasible set of initial robot configurations that maintain a fixed-grasp of the handle. Finally, the mobile base provides a significant relative increase in the average manipulability index at the end of the run.

Chapter 9

Conclusion

In this thesis, we tackle the problem of articulated object manipulation in an attempt to design a whole body control algorithm for a mobile base robot that would allow it to open different types of doors. We address the problem of designing and executing a door opening strategy after the initial grasp of the door handle has already been provided to us. After systematic decomposition of the complete problem into smaller functional modules, we propose a haptic feedback based approach, best described as a combination of optimization based planning and traditional feedback control. A solution has been proposed for each of the modules such that the robot could reliably open the door by additionally requiring only the information provided by the encoders mounted in the joints of the robotic arm. Several major milestones are achieved:

- We propose and develop an efficient initial direction estimation module capable of determining a direction in the Cartesian space that serves as a solid initialization for the iterative state estimation procedure. In addition, the proposed solution only requires performing short and simple jerk movements in the sampled candidate directions and recording the magnitude of the external wrench acting on the end effector.

- By combining the haptic feedback from the force/torque sensors mounted at each joint and the kinematic feedback provided by the joint encoders, an iterative state estimation procedure is developed that updates the current estimate of the direction in which the end effector can move freely.

- A velocity planner module that utilizes the additional DOFs provided by the mobile base is proposed in order to increase the robot's manipulability index during the door opening procedure while, at the same time, planning within the constraints imposed by the robotic platform. Furthermore, we explore three different optimization problem formulations and compare their performance in the simulation in order to pick the best one to be deployed on the real robot. We also show the influence of varying the mixing coefficient parameter m, the cut off frequency f_c, the cost coefficients c_1 and c_2 and the scaling parameter β on the overall performance of the closed loop model.

- After providing the proof of concept in the PyBullet simulation, a simulation of the low level control on the actual robot has been prepared in Gazebo. We organize the code in the Gazebo simulation such that it strongly resembles the one deployed on the real robot, which allows for fast prototyping. A state machine module is developed to allow the user to issue both the realtime and the non-realtime commands to the Franka arm.

- We show a successful implementation of the proposed whole body control method on a real robot that consists of a Franka robotic arm and a Ridgeback mobile base. We tested the algorithm on a drawer and a cabinet door model with both fixed and mobile base algorithms. In total, 60 successful runs for the drawer case were recorded and 20 for the cabinet door. The algorithm has proven to be robust to different grasping positions and orientations within the feasible set of initial robot configurations that maintain a fixed grasp of the handle and it has proven its ability to leverage the mobile base movement in order to increase the average manipulability index during the run.

In future work, extensive testing of the proposed algorithm should be performed on other real world door mechanisms which could require a substantial amount of additional parameter tuning in order to find a universal solution independent of the door model. Furthermore, the proposed solution in this thesis works in the scenario where the movement of the base has no limitations. It would be beneficial to also address the door opening task in a confined space such as the narrow hallway. It is to be expected that in this scenario, a significant number of additional constraints would have to be adequately designed in order to comply with the real time requirements of the application. Finally, integrating other modules that were out of scope of this thesis with our proposed solutions should be one of the main goals because only then are we going to have a complete, fully autonomous, door opening robot.

Bibliography

[1] M. Arduengo, C. Torras, and L. Sentis, "A versatile framework for robust and adaptive door operation with a mobile manipulator robot," *CoRR*, vol. abs/1902.09051, 2019.

[2] E. Klingbeil, A. Saxena, and A. Y. Ng, "Learning to open new doors," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2751–2757.

[3] L. Peterson, D. Austin, and D. Kragic, "High-level control of a mobile manipulator for door opening," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 3, 2000, pp. 2333–2338 vol.3.

[4] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," *CoRR*, vol. abs/1812.03201, 2018.

[5] S. Chitta, B. Cohen, and M. Likhachev, "Planning for autonomous door opening with a mobile manipulator," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 1799–1806.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, Jun. 1981.

[7] O. Brock, J. Trinkle, and F. Ramos, *Learning to Manipulate Articulated Objects in Unstructured Environments Using a Grounded Relational Representation*, 2009, pp. 254–261.

[8] S. Gu, E. Holly, T. P. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation," *CoRR*, vol. abs/1610.00633, 2016.

[9] B. Nemec, L. Zlajpah, and A. Ude, "Door opening by joining reinforcement learning and intelligent control," 07 2017, pp. 222–228.

[10] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *CoRR*, vol. abs/1702.02453, 2017.

[11] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *International Conference on Learning Representations*, 2018.

[12] L. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *Robotics and Automation, IEEE Transactions on*, vol. 14, pp. 166 – 171, 03 1998.

[13] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, 11 2016.

[14] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[15] I. A. Şucan and S. Chitta, "Motion planning with constraints using configuration space approximations," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1904–1910.

[16] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[17] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body MPC for a dynamically stable mobile manipulator," *CoRR*, vol. abs/1902.10415, 2019.

[18] M. Cefalo, E. Magrini, and G. Oriolo, "Sensor-based task-constrained motion planning using model predictive control ∗∗this work is supported by the eu h2020 project comanoid." *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 220–225, 2018, 12th IFAC Symposium on Robot Control SYROCO 2018.

[19] M. Bjelonic, D. Bellicoso, Y. Viragh, D. Sako, F. Tresoldi, F. Jenelten, and M. Hutter, "Keep rollin' - whole-body motion control and planning for wheeled quadrupedal robots," 01 2019.

[20] A. Dietrich, A. Albu-Schäffer, and G. Hirzinger, "On continuous null space projections for torque-based, hierarchical, multi-objective manipulation," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2978–2985.

[21] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," 01 2016.

[22] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *CoRR*, vol. abs/1405.7705, 2014.

[23] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[24] A. Jain and C. C. Kemp, "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*, 2010, pp. 1807–1814.

[25] H. Chen and J. Lee, "Path planning of 5-dof manipulator based on maximum mobility," *International Journal of Precision Engineering and Manufacturing*, vol. 15, pp. 45–52, 01 2014.

[26] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[27] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[28] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernandez Perdomo, "ros
control: A generic and simple control framework for ROS," *The Journal of Open Source Software*, vol. 2, no. 20, pp. 456 – 456, Dec. 2017.

[29] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[30] F. E. GmbH, "Franka control interface documentation."

[31] C. R. Inc., "Ridgeback indoor robot platform."

[32] I. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, p. 20150202, 04 2016.

[33] I. W. Selesnick and C. S. Burrus, "Generalized digital butterworth filter design," *IEEE Transactions on Signal Processing*, vol. 46, no. 6, pp. 1688–1694, 1998.

[34] L. Tan and J. Jiang, "Chapter 8 - infinite impulse response filter design," in *Digital Signal Processing (Third Edition)*, third edition ed., L. Tan and J. Jiang, Eds. Academic Press, 2019, pp. 315–419.

[35] M. Andersen, J. Dahl, and L. Vandenberghe, "cvxopt documentation."

[36] R. T. McGibbon, "quadprog documentation."