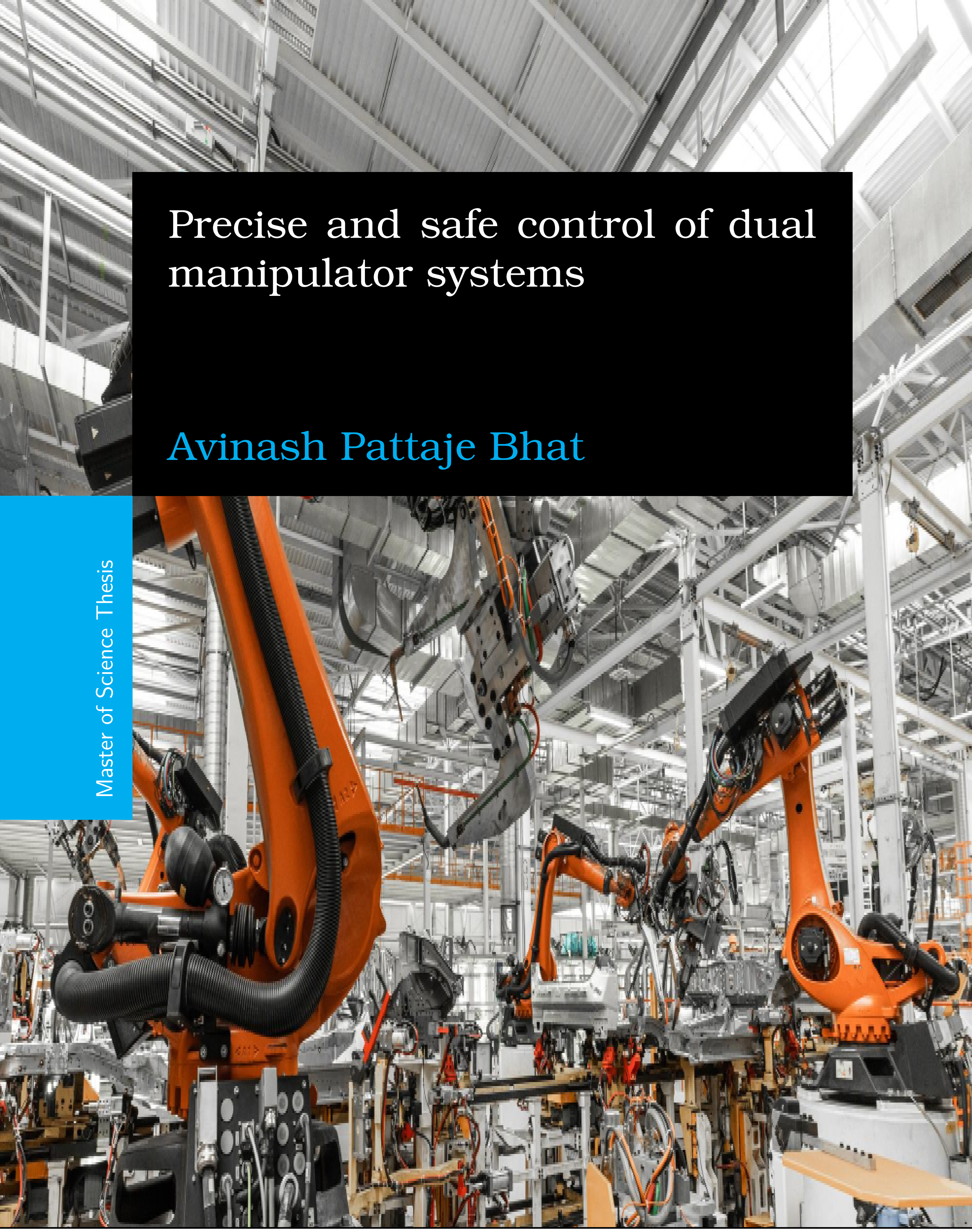


Precise and safe control of dual manipulator systems

Avinash Pattaje Bhat

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Precise and safe control of dual manipulator systems

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Avinash Pattaje Bhat

August 16, 2024

Faculty of Mechanical Engineering · Delft University of Technology

The work in this thesis was supported by Sioux Technologies B.V.. Their cooperation is hereby gratefully acknowledged..

Abstract

Robots have been used to automate repeatable tasks in industries for decades. With the rise of computing power, robots can now be used in a wider range tasks involving finer motion control, even ones which involve sharing a workspace with humans. An example of this is using a dual arm robot for quality control, as was demonstrated by BMW with their RoboCT quality control system. However, such systems require guarantees of safety to demonstrate reliable operation.

This thesis will delve into the question of how to compute and execute a motion plan for a dual robot system which has a guarantee of synchronized and collision-free motion. This thesis proposes using a hierarchical control approach to ensure task execution while satisfying collision avoidance and alignment constraints. The proposed approach is verified through a case study of using two manipulators to pick and place a ball.

Table of Contents

Acknowledgements

I would first like to thank my supervisor Dr. Manuel Mazo, Jr. for his assistance during the writing of this thesis. His suggestions on directions that this thesis could take (especially the suggestion to look at the work of Dr. Aaron Ames) were invaluable in arriving to the design proposed in this document. I would also like to thank my industry supervisor Dr. Caspar Gruijthuijsen for his guidance in the design process. His patience and guidance was invaluable in deciding how to make the design implementable in an industry setting and the assistance with understanding the intricacies of ROS 2 is greatly appreciated.

I would like to thank my friends in Systems and Controls. The times we spent together chatting in the sun between classes, having lunch together between work and getting coffee with the occasional foosball game could always be counted on to brighten up my day. To get to know all of you has been a privilege and undoubtedly one of the highlights at my time at TU Delft.

I would like to thank my friends from my Bachelor's for the random Discord calls we would have at 8 pm on a Saurday night. Catching up and reminiscing about the good times, and planning for all the future trips to come (which will certainly come to pass) were always a joy.

I would like to thank my colleagues at Sioux, for making me feel like a part of the family. The board game nights were events that I looked forward to every month, and the open bar certainly did not hurt.

Finally, I would like to thank my family. Their patience, understanding and love even in the most difficult of times were what gave me the strength to keep going, and to them I dedicate this capstone project of my MSc journey.

Delft, University of Technology
August 16, 2024

Avinash Pattaje Bhat

“In 30 years, a robot will likely be on the cover of Time magazine as the best CEO. Machines will do what human beings are incapable of doing. Machines will partner and cooperate with humans, rather than become mankind’s biggest enemy.”

— *Jack Ma*

Chapter 1

Introduction

Robots have been introduced into the industrial world since the 20th century. From Unimation, the first ever industrial robot which was deployed on the assembly line of a General Motors plant in New Jersey in 1961, robots have played an increasing role in the manufacturing industry, with their deployment being linked to an increase in productivity. In fact, robotics is seen as a key component for realizing the fourth industrial revolution.

This is not to say that the adoption of robots has been limited to manufacturing. Robots have also started being deployed in new roles and services like delivery, with tentative adoption seeing a rise during and post-COVID era. Robotics has even started to be adopted in the field of medicine, with it playing a significant role in the research into remote surgery procedures. A point of focus in all of this has been on the deployment of multi-arm systems. The limitations of a single arm setup have been increasingly become apparent in tasks like delicate assembly procedures, with the introduction of a second arm helping to circumvent these limitations.

1-1 Robotics in CT systems

While multi-arm robots were initially adopted in the manufacturing industry, due to the obvious advantages in performing repetitive tasks on an assembly line, they have also started to filter into more niche use-cases in our daily lives. One of these niche use-cases has been in implementing Computed Tomography (CT) systems. CT systems are used to reconstruct models of 3D objects by blending multiple 2D images from different perspectives. The reader is likely aware of the term from CT scans from hospitals, where X-ray and MRI machines are used to diagnose a patient's health conditions. Some readers might also be aware of the use of X-ray CT machines during surgery. Coincidentally, X-ray CT also happens to be an established method of non-destructive testing in the manufacturing industry. It allows manufacturers to make 3D reconstructions of their product and observe it from different cross-planes for flaws that might have arisen during the manufacturing process. This makes X-ray CT very valuable for quality control and testing as well.

Figure 1-1: Industrial robots deployed on an assembly line (Image credits: usertrmk on Freepik)

Most CT systems make use of a rotating structure with the X-ray source at one end and the detector at the other. The setup is such that the object to be imaged lies in the resulting focal plane. An example of this is the common C-arm structure used in most hospitals, as seen in Figure 1-2. However, such systems have a limitation in the size of the object that they can scan and are not very flexible. This problem was faced by the engineers at BMW when they were designing the BMW i3, as detailed in [1]. The flexibility issue was also raised in the context of an operating theater CT system in [2], as the size of the C-arm can prove to be a limiting factor in the amount of room surgeons have to maneuver during an operation.

While both of these research topics occurred in different places at different times, they both arrived at the same answer: overhaul the structure of the CT system. Instead of passively aligning the source and detector using a bulky structure, it was found that using a two-arm system to position them allowed for more flexibility in their respective use-cases. These redesigned systems have in fact been brought to market by different companies, as seen in Figures 1-3 and 1-4.

However, such a system presents its own set of unique challenges. Since the source and detector are no longer aligned by a rigid mechanical structure, it is necessary to actively keep them aligned during the entire imaging process. This requirement needs to be accounted for during both motion planning and execution. This problem leads us to the motivation for this thesis.

1-2 Motivation and outline

The problem of how to ensure successful CT scans inspires us to look into the problem of moving two robotic arms while keeping their End Effector (EEF)s aligned. The goal of this

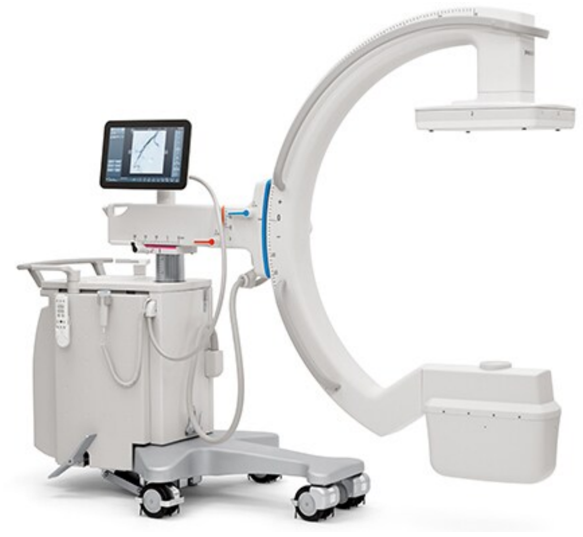


Figure 1-2: A classic C-arm X-ray CT system (image credits: Philips Healthcare)

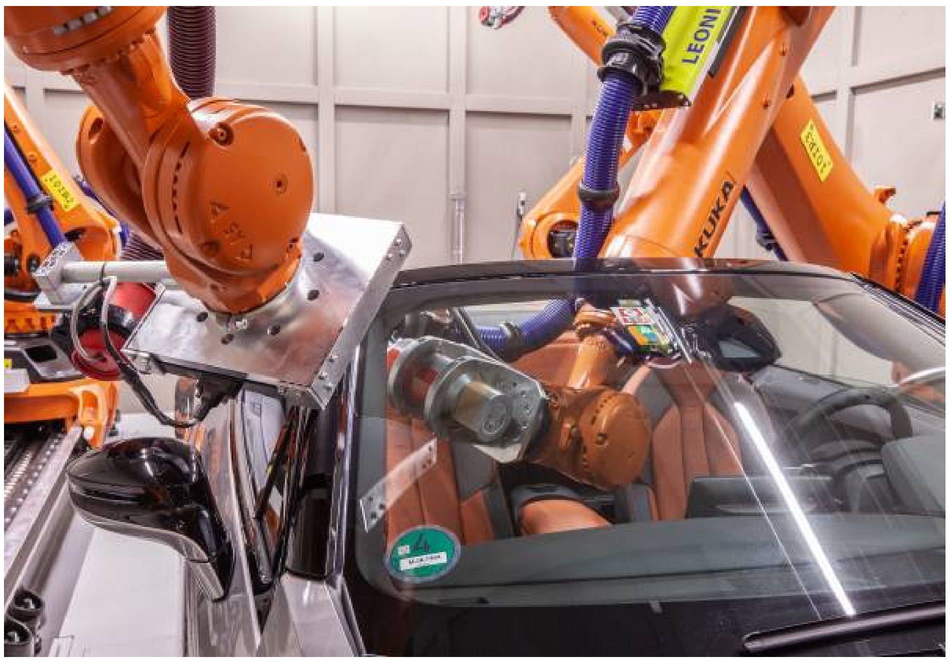


Figure 1-3: A robotic CT scanner for quality assurance developed by BMW [1, Figure 2]

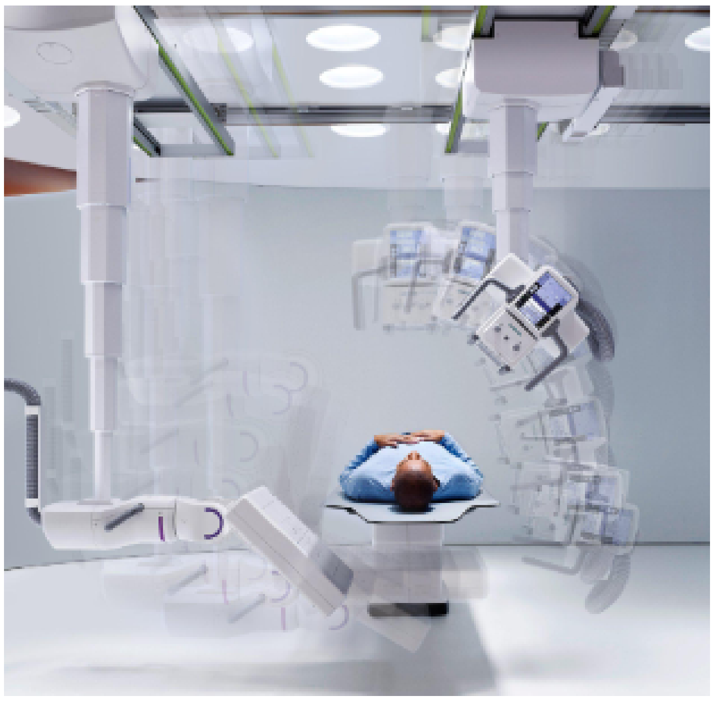


Figure 1-4: Dual-arm X-Ray system by Siemens Healthineers [3, Figure 1]

thesis is to develop a method to plan and execute a coordinated trajectory for two robotic arms using a kinematic model of the robots. There is a large body of literature that focuses on using force closures for controlling the trajectories of robot manipulators. This is an obvious approach when it comes to tasks like moving an object, as the object acts a link which closes the kinematic chain, thus allowing the forces applied by the EEF of one robot to to apply torques on the joints of the other.

However, this method is not very useful in the cases shown in [1–4]. In these cases, we wish for the system to behave as if it were a closed kinematic chain, even though in actuality the system is composed of two independent serial chains with no mutual link. This means that the trajectories need to be planned while ensuring that the relative pose constraints are satisfied (or the EEFs are aligned) for the entire duration of the trajectory.

While this problem is challenging enough as it is, it is further complicated when collision avoidance also has to be considered. Robots are increasingly sharing a common workspace with humans, especially in the scenarios considered in [2, 3, 5, 6]. This means the trajectories do not have to just be synchronized, but they have to be safe as well.

These points allow us to state the design requirements our trajectory planner must satisfy:

Requirement 1. *The trajectory planner must guarantee safety even in the presence of obstacles in the environment.*

Requirement 2. *The trajectory planner must ensure that the trajectory satisfies the relative constraints between the arms, subject to the satisfaction of Requirement 1.*

Requirement 3. *The planned trajectory must successfully execute the desired task (with the criteria for success based on the task, for e.g., like speed in a pick and place task, meeting given exposure times for CT, etc.), subject to the satisfaction of Requirements 1 and 2.*

With these three guiding requirements, we can start diving into the relevant literature. The next chapter will delve into the prior work that informed this thesis. The chapter will first give a brief introduction to kinematics, as well as how motion planning is implemented in popular motion planning frameworks. Then the idea of multi-rate hierarchical control schemes will be introduced, with the work done in an existing paper being used as an illustrative example. The last part of the chapter will give a brief introduction to some of the software tools used to implement the proposed architecture. The third chapter will delve into the design of a multi-rate hierarchical framework for the task at hand, as well as how to mathematically represent the requirements that were listed previously in this chapter. The fourth chapter will introduce a toy example used to demonstrate the performance of the proposed design, with the performance being contrasted with a naive Model Predictive Control (MPC) scheme. Finally, the fifth chapter will present the conclusion of this work, as well as possible avenues of research that could be pursued to improve this work.

Chapter 2

Trajectory planning concepts

This chapter will give a brief introduction to some ideas in literature, which will be then be used to in the next chapter to design a solution to the problems that were listed in Section 1-2. We start by briefly covering some preliminary information such as notation, the concept of spaces and kinematic as well as dynamical modelling. We then cover the concepts of sampling-based motion planning and hierarchical control using a contemporary work as an example. We conclude this chapter with a brief introduction to two important robot software tools.

2-1 Preliminaries

This section will briefly cover some of the basic information needed to comprehend the rest of the paper. We start with the notation that is used in the paper. Then, a brief overview is given of the different representations that can be used to represent robots. The last two subsections cover how a robot manipulator can be modelled.

2-1-1 Notation

The set of real, positive-real and non-negative numbers are denoted by $\mathbb{R}$, $\mathbb{R}^+$ and $\mathbb{R}_0^+$ respectively. The set of natural numbers is denoted by $\mathbb{N}$. The closed set of integers from a to b is denoted with $\mathbb{Z}_{a:b}$. In this work, both $[a\ b]^T$ and (a, b) are used to denote vertical concatenation of vectors a and b. These notations may be used interchangeably. The Euclidean norm is denoted by $||\cdot||$. The interior and boundary of a set $\mathcal{S}$ are denoted by $Int(\mathcal{S})$ and $\partial\mathcal{S}$ respectively. The distance from a point x to a set $\mathcal{S}$ is denote by $||x||_{\mathcal{S}} = \inf_{s\in\mathcal{S}} ||x - s||$. For any essentially bounded function $g : \mathbb{R} \to \mathbb{R}^n$, the infinity norm of g is denoted by $||g||_\infty = \text{ess}\sup_{t\in\mathbb{R}} ||g(t)||$.

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called Lipschitz continuous on $\mathcal{C} \subset \mathbb{R}^n$ if there exists a constant $L \in \mathbb{R}^+$ such that $||f(x_1)-f(x_2)|| \leq L||x_1-x_2||$ for all $x_1, x_2 \in \mathcal{C}$, and called locally Lipschitz continuous at a point $x \in \mathbb{R}^n$ if there exist constant $\delta, M \in \mathbb{R}^+$ such that $||f(x) - f(x')|| \leq M||x - x'||$ holds for all $||x - x'|| \leq \delta$. A function $\alpha_1 : [0, a) \to \mathbb{R}_0^+$ is said to belong to class $\mathcal{K}$ if it is strictly increasing. A function $\alpha_2 : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is said to belong to class $\mathcal{K}_\infty$ if it satisfies the conditions of a class $\mathcal{K}$ function and $\alpha_2(x) \to \infty$ as $x \to \infty$. A function $\alpha_3 : (-b, a) \to \mathbb{R}$ is said to belong to extended class $\mathcal{K}$ if it is strictly increasing and $\alpha_3(0) = 0$. Similarly, a function $\alpha_4 : \mathbb{R} \to \mathbb{R}$ is said to belong to extended class $\mathcal{K}_\infty$ if it satisfies the conditions of extended class $\mathcal{K}$ and $\alpha_4(x) \to -\infty$ as $x \to -\infty$ and $\alpha_4(x) \to \infty$ as $x \to \infty$. The notation $L_f g(\cdot)$ denotes the Lie derivative of the function $g(\cdot)$ with respect to the function $f(\cdot)$, i.e., $\frac{\partial g(x)}{\partial x} f(x)$.

2-1-2 Spaces

We will first take a quick look at the different spaces in which we can represent the state of the robot. Robots are mainly represented in either the *joint space* or the *task space*. The joint space refers to the space spanned by the combination of the joints, while the task space refers to the pose of the End Effector (EEF) in Cartesian space. An excellent visual example of this for a UR5 robot can be seen in Figure 2-1. In this scenario, the joint space representation of the robotic arm would be $(j_0, j_1, j_2, j_3, j_4, j_5)^T$, while the task space representation of the robot would be $(x, y, z, \alpha, \beta, \gamma)^T$, where α, β and γ refer to the Euler rotation angles.

It is possible to compute the EEF pose in the task space from the joint positions in the joint space using *forward kinematics*. Similarly, one could compute the position in joint space required to achieve some desired EEF pose in task space by using *inverse kinematics*. This is topic is explored in detail in [7, Chapter 2].

In this thesis, all our computations are performed exclusively in joint space. Therefore, we are interested in being able to obtain forward kinematic models which will allow us to compute our performance metrics based on the joint configurations of the robot.

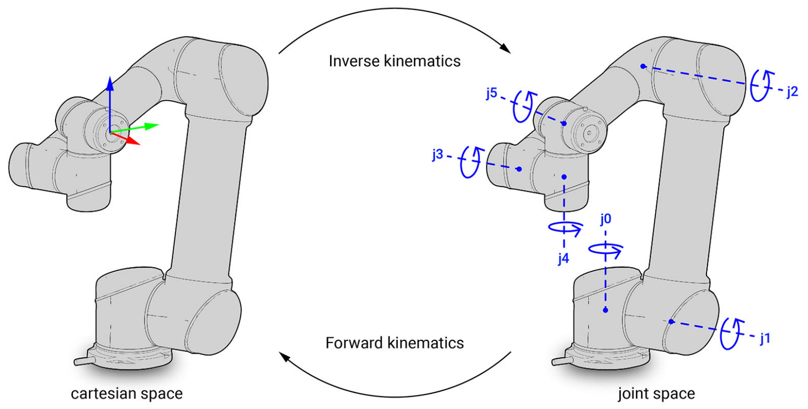


Figure 2-1: A representation of forward and inverse kinematics (Image credits: COMPAS FAB documentation)

2-1-3 Kinematic models

In order to be capable of representing the position of the EEF from the joint variables, it is desirable to be able to construct a closed form expression for the function mapping from the the joint space to the position of the EEF in Cartesian task space. This can be accomplished through the use of forward kinematics [7, Section 1.6].

A robotic manipulator is kinematically modelled as a sequence of perfectly rigid **links**, with these links connected together at **joints** where their idealized surfaces are in ideal contact without any clearance. These sequences of links and joints can be represented as a series of coordinate frames. For a simple example, consider a two link setup shown in Figure 2-2. Assume that Frame 0 is stationary and fixed to one end of Frame 0, and Frame 1 is attached to one end of Link 1. Link 1 and Frame 1 can rotate about the z axis (the blue axis pointing out of the plane).

The transformation from Frame 0 to Frame 1 can be decomposed into two transformations, as seen in Figure 2-3. Assume that there is another frame (referred to as Frame 1') which shares an origin with Frame 1. By moving Frame 0 along its x axis (the red arrow) by a fixed distance, it will overlap with Frame 1'. Then, by rotating Frame 1' about its z axis, it will overlap with Frame 1. By combining these two operations (first moving a fixed distance along one axis and then rotating around another), we obtain a *screw transformation* between frames 0 and 1.

Mathematically, if the rotation was represented using a rotation matrix ${}^{0}R_1 \in \mathbb{R}^{3\times3}$ (${}^{1'}R_1$ in Figure 2-3), and the displacement was represented with a displacement vector ${}^{0}t_1 \in \mathbb{R}^3$ (${}^{0}t_{1'}$ in Figure 2-3), then the transformation matrix ${}^{0}T_1$ can be written as

$$ {}^{0}T_1 = \begin{bmatrix} {}^{0}R_1 & {}^{0}t_1 \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{2-1}$$

Given that the rotation matrix would be a function of the angle rotated (say q), then the transformation and rotation matrices can be written as ${}^{0}T_1(q)$ and $0^R_1(q)$ respectively. The displacement vector is a constant since the link is assumed to be rigid.

This idea can then be extended to the modelling of a robotic manipulator. Imagine that each of the six joints in the UR5 robot as seen in Figure 2-1 has a reference frame attached to it, with the axis of rotation being the z axis. Also imagine a stationary reference frame w fixed to the base of the robot. This reference frame can be transformed to the EEF frame seen in Figure 2-1 by applying a sequence of screw transformations from each reference frame to the next.

$$ {}^{w}T_{EEF} ={}^{w} T_{j0}(j0)\ {}^{j0}T_{j1}(j1)\ {}^{j1}T_{j2}(j2)\ {}^{j2}T_{j3}(j3)\ {}^{j3}T_{j4}(j4)\ {}^{j4}T_{j5}(j5)\ {}^{j5}T_{EEF} $$

Note how this transformation would be a function of the rotations about each joint (i.e., the joint values). This transformation matrix is the forward kinematics. For further reading, the reader is advised to look at [7, Section 2.3].

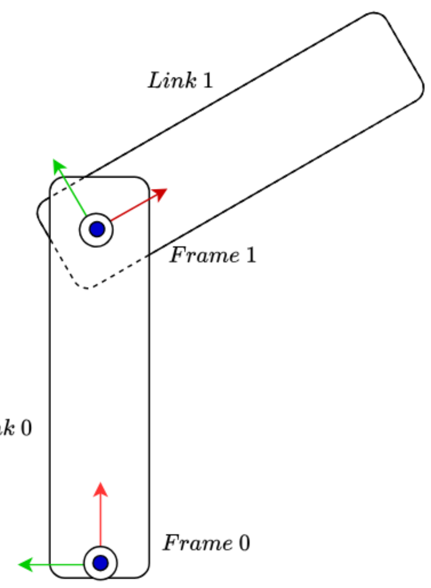


Figure 2-2: A simple two-link joint

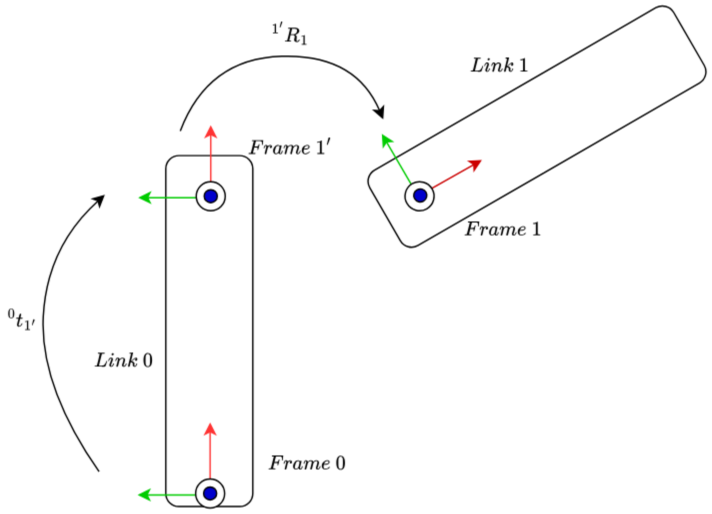


Figure 2-3: A two-link setup in an exploded view

2-1-4 Dynamic models

The previous section dealt with how the physical structure of the robotic manipulator could be modelled. However, it did not address how the robot's inertia affects the dynamics. This is not a problem if the manipulator comes with tracking controllers that are capable of reliably tracking the desired velocity and acceleration profiles, as assumed in Chapter 4. However, in the absence of such controllers, it would be necessary to account for these inertial effects.

Such models can be derived either using the Newton-Euler approach (writing the different forces and torques for each link using Newton's Second Law) or by using the Lagrangian approach (which uses the Lagrangian formula based on potential and kinetic energy). Both methods lead to the same model. This model, as described in [7, Section 3.3], is written as

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_g(q) = \tau \tag{2-2}$$

This model is derived from Newton's Second Law, by balancing the torques. $H(q)\ddot{q}$ refers to the torque components created by inertial affects, $C(q, \dot{q})\dot{q}$ refers to torques computed by Coriolis effects and $\tau_g(q)$ refers to torques produced by gravitational effects. The net sum of these internal torques of the system is equal to the external torque applied τ. Naturally, more terms can be added to account for extra forces (for e.g. friction, air resistance, etc.).

With this, we have enough information about how a single manipulator can be modelled in both joint space and Cartesian EEF space. With this knowledge, we can now delve into the motion planning problem, which is covered in the next section.

2-2 Sampling-based motion planning

This section deals with how trajectory planning is implemented in general (sampling-based) motion planning frameworks. The most common definition of the planning problem is the Piano Mover's problem, defined below and visualized in Figure 2-4.

Definition 2.1 (The Piano Mover's Problem [8]). *Given*

- *A Cartesian world $\mathcal{W}$.*
- *A semi-algebraic obstacle region $\mathcal{O} \subset \mathcal{W}$.*
- *A semi-algebraic robot defined in $\mathcal{W}$.*
- *A configuration space (or state space) $\mathcal{C}$, representing the set of all possible configurations (or states) of the robot. This set can be further partitioned into the set of collision-free configurations $\mathcal{C}_{free}$ and configurations in collision with the obstacles $\mathcal{C}_{obs}$.*
- *A configuration $q_I \in \mathcal{C}_{free}$ designated as the initial configuration.*
- *A configuration $q_G \in \mathcal{C}_{free}$ designated as the goal configuration.*

The goal is to compute a continuous path $\pi : [0, 1] \to \mathcal{C}_{free}$, such that $\pi(0) = q_I$ and $\pi(1) = q_G$.

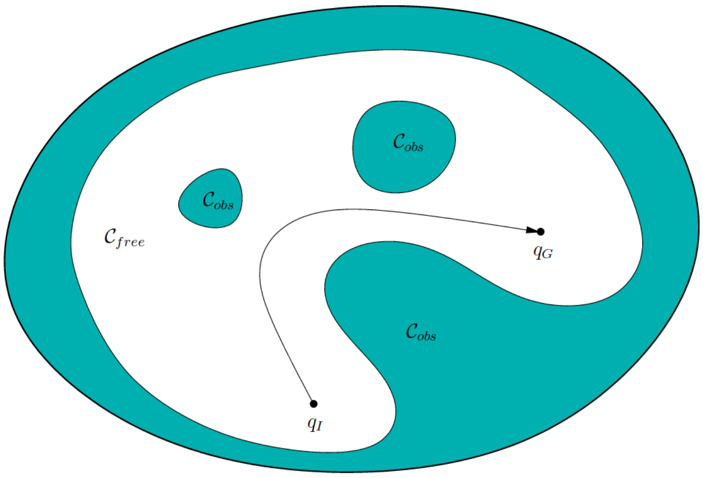


Figure 2-4: A visualization of the planning problem [8, Figure 4.11]

The configuration space $\mathcal{C}$ is analogous to a state space $\mathcal{X}$, with the configurations q_I, q_G being analogous to states x_{init}, x_{goal}. The world $\mathcal{W}$ is analogous to the set of output states $\mathcal{Y}$, with the pose of the robot in the world being the output y. We will use the state space notation for the rest of this thesis.

The method of choice to solve most motion planning problems tend to be sampled based motion planners. Such motion planners avoid explicitly constructing the sets $\mathcal{X}_{obs}$ and $\mathcal{X}_{free}$. Instead, they use a graph constructed using dense-sampling to represent the state space $\mathcal{X}$. How the graph itself is constructed can vary between methods (online in the case of Rapidly exploring Random Trees (RRT) [9] versus offline persistent graphs like in Probabilistic Roadmaps (PRM) [10]). Most algorithms follow the following template for graph construction

Algorithm 2.1 (Graph construction template [8])**.** *Initialize an undirected graph $\mathcal{G}(V, E)$, where V is a set of vertices initialized with at least one vertex x_{init}, x_goal or both, and E is an empty set of edges. Let there be a termination condition T_1, which denotes that the graph $\mathcal{G}$ encodes a solution path and a termination condition T_2, which denotes the algorithm has failed. Repeat the steps below infinitely.*

1. *Select a vertex $x \in V$ for expansion.*
2. *Sample some new state in the collision-free state space $x_{new} \in \mathcal{X}_{free}$. Attempt to construct a local path $\pi_s : [0, 1] \to \mathcal{X}_{free}$ such that $\pi_s(0) = x$ and $\tau_s(1) = x_{new}$. If this step fails, return to step 1.*
3. *Insert τ_s in E as an edge from x to x_{new}. If x_{new} does not exist in V, it is added as well.*
4. *Check if either T_1 or T_2 is satisfied. If yes, exit and return the graph $\mathcal{G}$ with the corresponding result. Else, return to step 1.*

In this manner, the original continuous state space can be represented using a discrete data structure. Once the graph is constructed, a solution path on the graph can be found using algorithms such as Dijkstra and A^*.Such a solution path would be a sequence of states $\bar{x}$ and the corresponding sequence of edges $\bar{\pi}$. It is plain to see that this series of edges would form a path that solves the Piano Mover's Problem as defined in Definition 2.1. Note that this path may not necessarily be smooth, and in some algorithms such as PRM may not even account for the dynamics of the system.

Such algorithms are favoured due to the fact that they work well even in fairly large dimensional state spaces. Another nice property that sampling based motion planners have is the notion of *probabilistic completeness*. This means that as long as the algorithm keeps sampling states, it will eventually find a solution to the planning problem, provided one exists. Once a plan is generated, in can be used in one of three ways, as noted in [8, Section 1.4.3].

- The plan can be used to generate a control law that can be loaded into the robot, after which the robot acts autonomously.
- The plan can be refined by passing them to optimizing planning algorithms such as Covariant Hamiltonian Optimization for Motion Planning (CHOMP)[11] or Stochastic Trajectory Optimization for Motion Planning (STOMP)[12]. This can be done either to make it more efficient or to account for more problem aspects such as system dynamics.
- The plan can be used in a hierarchy, with the plan imagined as a subroutine in a larger plan.

It is the third approach that is of interest to us, and will be covered in the next section.

2-3 Multi-rate hierarchical control schemes

Hierarchy in control theory is seen as a way to break down a large (and computationally difficult) control problem into more manageable sub-problems at different levels. To the best of the author's knowledge, the concept of hierarchical control was first defined in [13]. The most intuitive definition that the author is aware of was provided by Mesarovic in [14].

According to Mesarovic, any system could be described as a multilevel hierarchical system if it satisfied three conditions:

- There is a vertical decomposition into subsystems.
- The subsystems higher up in the hierarchy can intervene in the functioning of subsystems lower in the hierarchy.
- The subsystems lower in the hierarchy can send feedback of their own performance to subsystems higher in the hierarchy.

This can be a very broad interpretation, as it can encompass both temporal hierarchies as seen in [13], as well as spatial hierarchies reminiscent of distributed control as seen in the works [15, 16].

We are interested in the temporal multilayer hierarchical control paradigm, where the controller is split across multiple layers working at different frequencies, as this is the hierarchy

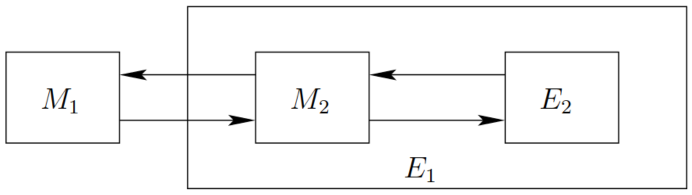


Figure 2-5: A hierarchical planning model [8, Figure 1.20]

referred to in [8] and seen in Figure 2-5. The planning machine M_1 interacts with its environment E_1, but the environment itself is composed of a planning machine M_2 interacting with its own environment E_2, with no limit on the depth of this recursion.

Such a strategy is common when implementing a decision making process for autonomous systems, using high level planners with controllers for planning and execution of joint trajectories as shown in the works of [17–20]. A contemporary example of this is the work of [17], which is a big inspiration for the design proposed in Chapter 3 to address the problems listed in Section 1-2. In this design, we see that there is a high level decision maker that abstracts the world as a finite-state machine, a Model Predictive Control (MPC) that treats the system as a discrete time linear system and a Control Barrier Function (CBF) based controller that ensures the real-time safety of the system.

2-3-1 High-level decision making

At the highest layer, the system is usually abstracted as a Finite State Machine (FSM). As an example, in the work of [17] the environment is partitioned uniformly into a grid of cells. A high-level decision maker uses information about the current state to reason about the current grid cell of the FSM the system is currently in. It then determines the next grid cell (as well as the next state in that grid cell) that the system should be driven to.

This can be seen as being analogous to how traditional motion planning algorithms work, as described in Section 2-2. This similarity can be seen most clearly in how the motion planning algorithms SPArse Roadmap Spanner (SPARS) and SPARS2 [21] work. The goal

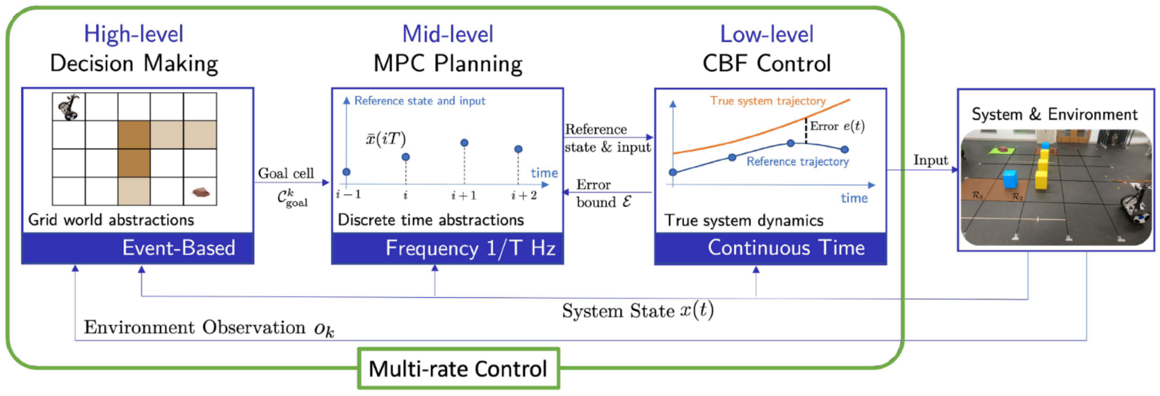


Figure 2-6: A multirate control architecture [17, Figure 2]

in constructing the roadmap spanner (a subgraph of a roadmap) is that no state should be more than a distance Δ from a vertex of the spanner. In other words, the spanner implicitly partitions the free state space, such that each partition is represented by a vertex on the spanner (in the terminology of [21], a partition is referred to as the visibility region of a vertex).

Some works like [19, 22] augment this graph representation with an automata, which allows for more rich motion plan requests like visiting specific regions of the state space in a certain order while avoiding obstacles. In essence, these layers output parameters (whether it be in the form of reference trajectories, state constraint sets, etc.) which inform the operation of the middle layer.

The highest layer is essentially a reference generator, used to autonomously generate a sequence of intermediate states, tracking which it is possible to drive the system from the start state to the goal state. In traditional trajectory planning, the entire sequence of control inputs is calculated in a single go. In the hierarchical control scheme, this reference can then be tracked by some tracking feedback controller in order to drive the system to the goal state. This reference sequence of intermediate configurations (henceforth referred to as **waypoints**) can either be generated in one go (as usually happens with motion planning algorithms), or in an event-driven manner with a new waypoint being generated when the current waypoint is reached (as happens in [17]). As shown in [23], this reference can be replaced by a user-generated reference as well.

2-3-2 Mid-level Model Predictive Control planning

The middle stage of the motion planning framework makes use of MPC, an optimal control strategy that minimizes some trajectory cost over a finite horizon, subject to certain constraints on the states and control inputs of the system.The standard structure of an MPC problem for a discrete time Linear Time Invariant (LTI) system can be observed in Problem 2-3. It has been proved that the LTI MPC can be made asymptotically stable [24, Theorem 2.19].

$$\begin{aligned} \min_{\mathbf{u}} \ & \sum_{k=0}^{N-1} l(x(k),u(k)) + V_f(x(N)) \\ s.t. \ & x(k+1) = Ax(k) + Bu(k) \quad \forall k \in \mathbb{Z}_{0:N-1} \\ & x(k) \in \mathcal{X} \quad \forall k \in \mathbb{Z}_{1:N} \\ & u(k) \in \mathcal{U} \quad \forall k \in \mathbb{Z}_{0:N-1} \\ & x(N) \in \mathcal{X}_f \\ & x(0) = x_0 \end{aligned} \tag{2-3}$$

Given that systems usually contain some level on nonlinearity, the LTI MPC model cannot be used directly. Fortunately, the existing literature has substantially explored methods on how to account for nonlinearities in the MPC design. These nonlinearities can accounted for by linearizing the system about a candidate trajectory, either through the use of models [17, 25, 26] or by using data-driven approximations [20, 27].

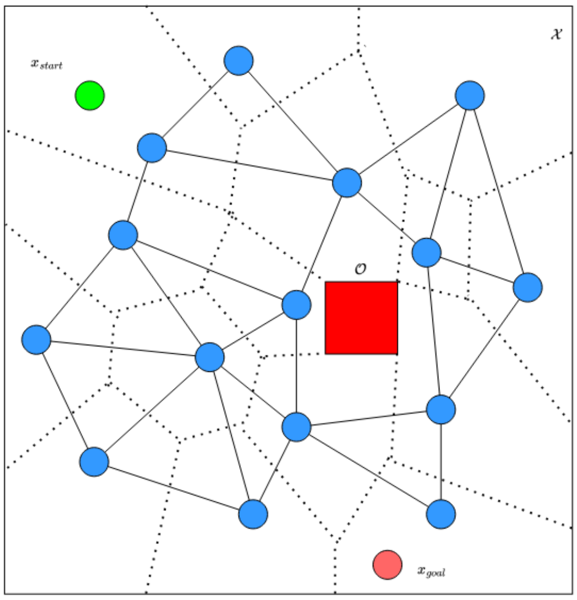


Figure 2-7: Visualization of how SPARS implicitly partitions a state space

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \; & \sum_{k=0}^{N-1} l(x(k),u(k)) + V_f(x(N)) \\
s.t. \quad & x(k+1) = f(x(k),u(k)) \quad \forall k \in \mathbb{Z}_{0:N-1} \\
& x(k) \in \mathcal{X} \quad \forall k \in \mathbb{Z}_{1:N} \\
& u(k) \in \mathcal{U} \quad \forall k \in \mathbb{Z}_{0:N-1} \\
& x(N) \in \mathcal{X}_f \\
& x(0) = x_0
\end{aligned} \tag{2-4}$$

Alternatively, the MPC can be written with the nonlinear system dynamics as in Problem 2-4, and the resulting non-convexity of the optimization problem can be handled by making use of a non-convex optimization algorithm (like Sequential Quadratic Programming (SQP) and interior-point methods). Such a formulation is not guaranteed to give the best solution, but it can provide a "better" solution. In such a scenario it is of interest to determine if such suboptimal solutions can still lead to stability of the system in closed loop. For the sake of brevity, the proposition is stated below as-is, taken from the source material. The assumptions and the corresponding algorithm are described in more detail in Appendix A.

Proposition 2.1 (Stability of Suboptimal MPC [24]). *Suppose Assumptions A.1, A.2 and A.4 are satisfied (see Appendix A), and that $l(x,u) \geq \alpha_l(|(x,u)|)$ (where α_l is an extended class $\mathcal{K}_\infty$ function) for all $(x,u) \in \mathcal{Z}$, and $\mathcal{X}_f = \{x \in \mathbb{R}^n \mid V_f(x) \leq b\}$, for some $b > 0$. Then the function $V_N(s)$ is a Lyapunov function in the set $\tilde{\mathcal{S}}_N$ for the closed loop system as defined in Equation A-4 under Algorithm A.1 (see Appendix A). Therefore, the origin is asymptotically stable in $\tilde{\mathcal{S}}_N$.*

The long and short of Proposition 2.1 is that even when it is not guaranteed that the solution found will be globally optimal, even a locally optimal solution can ensure that the closed loop system is asymptotically stable by using an appropriate (feasible) initial guess to warmstart the solver.

One of the most difficult parts of MPC design is constructing the terminal set $\mathcal{X}_f$. A terminal set that is too small can easily render the Nonlinear Model Predictive Control (NMPC) problem infeasible. However, the terminal set cannot be so large that it is no longer control invariant. The work in [28] investigated whether the NMPC control problem could be rendered stable while removing the terminal constraint. They discovered that by weighting the terminal cost of the cost function with a scalar coefficient β, one could induce a region of attraction around the origin. If the value of β was made large enough, that would render the closed loop system stable. This leads us to the final Proposition for MPC that we use in this work, taken from [24]

Proposition 2.2 (MPC stability without terminal constraint [24])**.** *The origin is asymptotically or exponentially stable for the closed loop system $x^+ = f(x, \kappa_N^\beta(x))$ with a region of attraction Γ_N^β. The set Γ_N^β is positive invariant for $x^+ = f(x, \kappa_N^\beta(x))$.*

[28] shows that this region of attraction grows with β, i.e., for $\beta_2 \geq \beta_1$, the corresponding regions of attractions obey the relation $\Gamma_N^{\beta_1} \subseteq \Gamma_N^{\beta_2}$. It is also proved that for any $x \in \mathcal{X}_N$ steerable to the terminal set $\mathcal{X}_f$, there exists a β such that $x \in \Gamma_N^\beta$. Therefore, weighting the terminal penalty can be used to ensure stability even in a suboptimal MPC controller. However, it is also noted in [29, Remark 2] that increasing the region of attraction by increasing the weight of terminal cost is at the expense of a loss of optimality (since the cost function used in the NMPC is a worse approximation of the cost function of the true optimal controller with terminal constraints).

2-3-3 Low-level Control Barrier Function control

One of the most popular methods has been to use a form of projection based on the CBF to ensure that the control input applied to the system never leads to any unsafe behaviour. The most favoured method to enforce such behaviour, evidenced by its use in [17, 20, 23, 30–34] in order to enforce collision avoidance in robots. A CBF-QP has also been used in [35] to enforce avoiding singular configurations of a robot manipulator. This demonstrates how flexible the CBF-Quadratic Programming (QP) is: provided that the system dynamics have a certain structure and a CBF can be found to demarcate the safe set, a CBF-QP can be constructed to ensure the safety of the controlled system.

Definition 2.2 (Forward invariance of a set [31])**.** *Under the locally Lipschitz system dynamics $\dot{x} = f(x)$, the set $\mathcal{C}$ is forward invariant if for every $x_0 \in \mathcal{C}$, $x(t) \in \mathcal{C}$ for $x(0) = x_0$ and all $t \in [0, \tau_{max})$. The system is safe with respect to the set $\mathcal{C}$ if the set $\mathcal{C}$ is forward invariant.*

$$\dot{x} = f(x) + g(x)u \tag{2-5}$$

Throughout this thesis, it is assumed that the system under consideration is nonlinear control affine as seen in Equation 2-5 (which is reasonable considering that manipulator dynamics can indeed be written in that form). f and g are assumed to be locally Lipschitz, with $x \in \mathcal{D} \subset \mathbb{R}^n$ and $u \in \mathcal{U} \subset \mathbb{R}^m$.

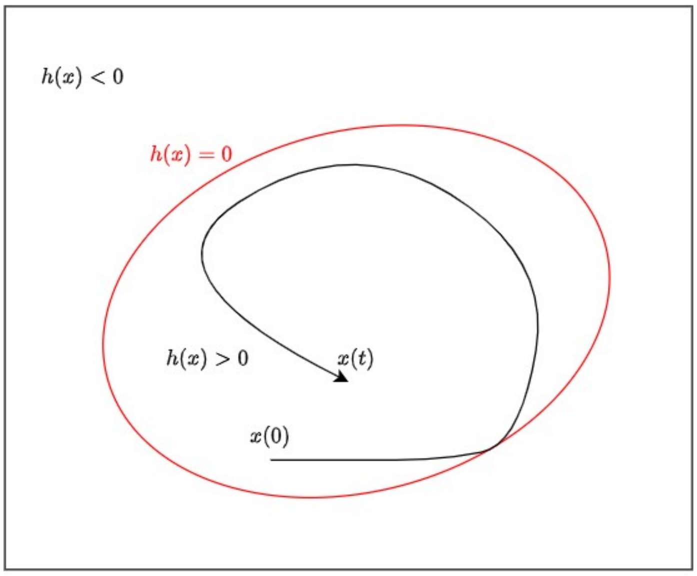


Figure 2-8: A visualization of set invariance

To borrow the words of [31], there are many approaches that use a "Lyapunov-like" approach to ensure the forward invariance of a set. Much like how Lyapunov theory searches for an appropriate Lyapunov function candidate and then attempts to bounds the time derivative of the function from above by a class $\mathcal{K}$ or class $\mathcal{K}_\infty$ to show asymptotic stability, barrier methods search for an appropriate barrier function candidate and then attempts to bound the time derivative of the barrier from below with an appropriate monotonic function.

Definition 2.3 (Control barrier function [31])**.** *Let $\mathcal{C} \subset \mathcal{D} \subset \mathbb{R}^n$ be the superlevel set of a continuously differentiable function $h : \mathcal{D} \rightarrow \mathbb{R}$, then h is a CBF if there exists an extended class $\mathcal{K}_\infty$ function α such that for the control affine system as defined in [31]*

$$\sup_{u \in \mathcal{U}} [L_f h(x) + L_g h(x) u] \geq -\alpha(h(x))$$

for all $x \in \mathcal{D}$.

Proposition 2.3 (Safety of a control system [31])**.** *Let $\mathcal{C} \subset \mathbb{R}^n$ be a set defined as the superlevel set of a continuously differentiable function $h : \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R}$. If h is a CBF on D and $\frac{\partial h}{\partial x} x \neq 0$ for all $x \in \partial\mathcal{C}$, then any Lipschitz continuous controller $u(x) \in K_{cbf}(x)$ for the control affine system as defined in Equation 2-5 renders the set $\mathcal{C}$ safe. Additionally, the set $\mathcal{C}$ is asymptotically stable in $\mathcal{D}$.*

As demonstrated in [34], the CBF-QP based controller can be used as a minimally invasive controller that supports a nominal (potentially unsafe) controller k_{des}. The CBF-QP based safety filter ensures the smallest possible deviation from the nominal control value, while ensuring that the system trajectory remains bounded in the safe set.

As noted in [31, Remark 6], while using a CBF based controller will not formally leave the safe set C, in practical implementation noise and modeling errors could lead to the forward invariance of C being violated. However, the CBF also ensures that $\mathcal{C}$ is asymptotically stable in $\mathcal{D}$, which ensures that even in the case of violations, the system trajectory is steered back into $\mathcal{C}$. Specifically, the function h induces a function $V_{\mathcal{C}} : \mathcal{D} \rightarrow \mathbb{R}_0^+$ defined by

$$V_{\mathcal{C}}(x) = \begin{cases} 0 & if\ x \in \mathcal{C} \\ -h(x) & if\ x \in \mathcal{D} \backslash \mathcal{C} \end{cases} \tag{2-6}$$

Proposition 2.4 (Asymptotic stability of a barrier function [29])**.** *Let $h : \mathcal{D} \rightarrow \mathbb{R}$ be a continuously differentiable function defined on an open set $\mathcal{D} \subseteq \mathbb{R}^n$. If h is a CBF for the control affine system defined as in Equation 2-5, then the set $\mathcal{C}$ defined by h is asymptotically stable. Moreover, the function V_C in Equation 2-6 is a Lyapunov function.*

A nice property of the induced Lyapunov function is that it allows us to infer the kind of effect that noise and modelling errors would have on the system, as well as its effects on our desired property of set invariance. These inferences were made in [29] for the cases when the noise was bounded as well as unbounded but vanishing.

Proposition 2.5 (Robustness of a barrier function [29])**.** *Under the assumptions of Proposition 2.4, the following statements hold:*

- *There exists $\varepsilon \in \mathbb{R}_0^+$ and a class $\mathcal{K}$ function $\sigma : [0, \varepsilon] \rightarrow \mathbb{R}_0^+$ such that for any continuous function $g_1 : \mathbb{R}^n \rightarrow \mathbb{R}^n$ satisfying $||g_1(x)|| \leq \sigma(||x||_{\mathcal{C}})$ for $x \in \mathcal{D} \backslash Int(\mathcal{C})$, the set $\mathcal{C}$ is still asymptotically stable for the system $\dot{x} = f(x) + g_1(x)$ describing the effect of a disturbance modeled by $g_1(\cdot)$ on the system defined in Equation 2-5.*
- *There exists a constant $k \in \mathbb{R}^+$ and class K function γ such that the set $\mathcal{C}_{\gamma(||g_2||_\infty)} \subseteq \mathcal{D}$ is locally asymptotically stable for the system $\dot{x} = f(x) + g_2(t)$ describing the effect of a disturbance modeled by g_2, and satisfying $||g_2||_\infty \leq k$, on the system defined in Equation 2-5.*

As mentioned in [34], the CBF-QP acts as a sort of safety filter on the control action generated by the nominal controller, only acting when the nominal control input would end up violating the CBF condition from Definition 2.3. This line of reasoning explains why when the nominal controller is proved to be asymptotically stable, it is taken to be trivially true that the controller resulting from cascading with a safety filter is also asymptotically stable.

2-4 Simulation tools

The following tools are used to implement the planner as well as generate a simulation of the systems. This section will give a brief overview of what utility each of the tools provide, as well as provide some insight into what each of the tools does under the hood.

2-4-1 Robot Operating System 2

Robot Operating System (ROS) [36] is an open-source communication layer that sits on top of the actual operating system of the computer. First introduced in 2007 by Willow Garage, ROS has quickly grown to become one of the most popular frameworks for designing robot software in hobbyist and academic communities. This popularity can be attributed to the fact that it allowed roboticists to use software written by other roboticists, meaning there was no longer a need to reinvent the wheel to enable certain functionalities. Recently, ROS was rebuilt from the ground-up in order to address certain shortcomings that hindered its widespread adoption in industry, namely security, reliability in non-traditional environments and large scale support for embedded systems. This led to the release of ROS2 [37] based on Data Distribution Service (DDS), which is also the framework used in this work.

While the underlying implementations may differ, both ROS and ROS2 fundamentally abstract the robot software in a similar way. The software that runs on the robot is split across multiple programs (called **nodes**), and these programs communicate using standardized interfaces (called **topics**). This is visualized in Figure 2-9. The nodes work in an event-based manner, processing new data when it becomes available. The advantage of abstracting the software in this manner is that it allows each node to be self-contained, meaning that nodes only process information that is available to them on their own topics, without having to consider how that information was obtained. This allows easily integrating the work of other programmers with minimal hassle, irrespective of the coding language the nodes are written in.

2-4-2 MoveIt2

One of the most popular motion planning software packages for serial robotic manipulators built on ROS is MoveIt [38]. It is a motion planning framework that works with multiple existing motion planners using a plugin interface. This allows users to switch between using motion planners by simply changing the name of the motion planner they would like to use, and MoveIt handles any required changes to interface with the motion planner in the background. With the release of ROS2, the developers released MoveIt2, which migrated some of the features of MoveIt to the new architecture while adding on some new features as well.

One of the new features that was added with MoveIt2 was the hybrid planning framework [39]. This feature was meant to extend motion planning algorithms to accomodate dynamic local constraints (such as moving obstacles or adapting to uncertainties in the task). This is accomplished by augmenting the output of the motion planning algorithm (referred to as a global planner) with the action of a local planner that tracks the global plan while accounting for changes in the local environment. The architecture is as seen in Figure 2-10. The reader may notice that the thought process involved is very similar to the idea of hierarchical control introduced in Section 2-3.

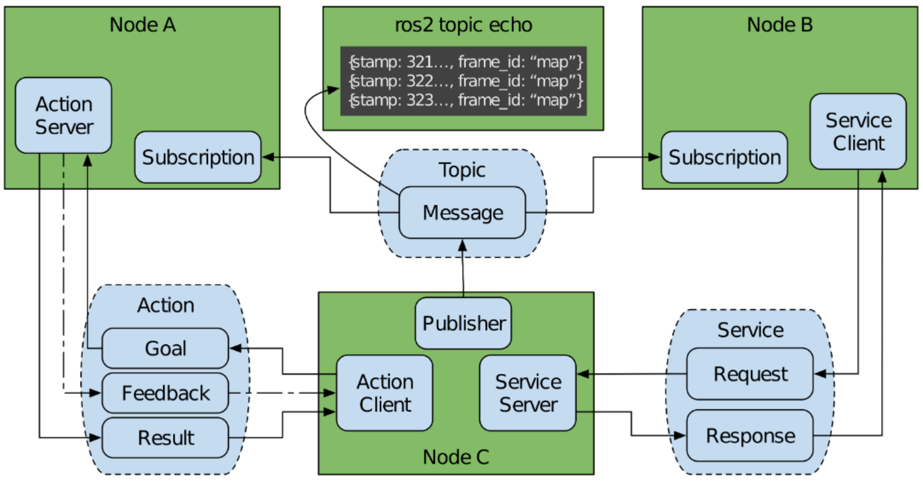


Figure 2-9: The different node interfaces available in ROS2 [37, Figure 1]

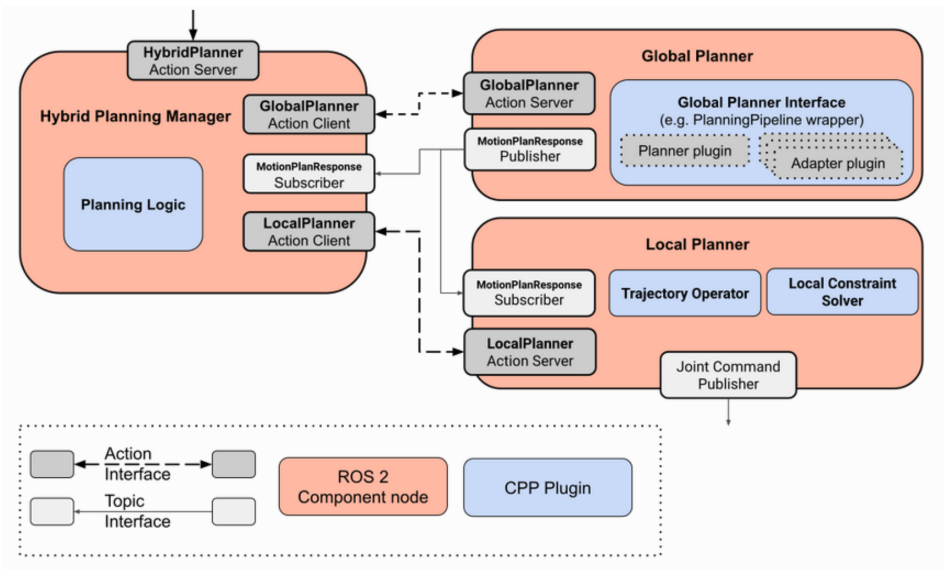

HybridPlanner Action Server
Hybrid Planning Manager
GlobalPlanner Action Client
MotionPlanResponse Subscriber
Planning Logic
LocalPlanner Action Client
Global Planner
GlobalPlanner Action Server
MotionPlanResponse Publisher
Global Planner Interface (e.g. PlanningPipeline wrapper)
Planner plugin
Adapter plugin
Local Planner
MotionPlanResponse Subscriber
Trajectory Operator
Local Constraint Solver
LocalPlanner Action Server
Joint Command Publisher
Action Interface
Topic Interface
ROS 2 Component node
CPP Plugin

Figure 2-10: The hybrid planning framework available in MoveIt2 [39]

Chapter 3

Proposed trajectory planning architecture

In this chapter, a trajectory planning framework is proposed to address the problem that was stated in Section 1-2. The proposed trajectory planner is as seen in Figure 3-1. The reader will notice that the design is quite similar to the work in [23] and [17], albeit simpler due to the simpler nature of the problem this paper is attempting to solve. The highest layer attempts to construct a reference motion plan from start to finish, the middle layer incorporates knowledge of any changes in the local environment since the construction of the initial motion plan and focuses on driving the system to the desired goal, while the lowest layer focuses only on ensuring that no safety conditions are violated. The main contribution in this section is how to formulate Control Barrier Function (CBF) conditions to formally guarantee alignment of the End Effector (EEF)s.

3-1 Global planner

The first layer is used to generate a reference motion plan composed of sequence of joint configurations from the starting state to the goal state. This layer will receive a snapshot of the environment, from which it will partition into a graph, of which the start and the goal state are vertices. This graph is then queried for a sequence of edges (which is consequently a sequence of vertices) to be traversed to reach the goal state. This layer can use conventional motion planning architectures like Rapidly exploring Random Trees (RRT) [9] and Probabilistic Roadmaps (PRM) [10] for simpler motions like a pick-and-place task (which is the example used in this work). For more complex motions like Computed Tomography (CT) imaging, where the source and the detector would have to take multiple different poses around the object to have enough images for 3D reconstruction, one could use more complex planning algorithms based on symbolic logic, like the work in [19]. In essence, the top layer generates a sequence of intermediate robot configurations (vertices on the graph) which acts as an initial trajectory plan.

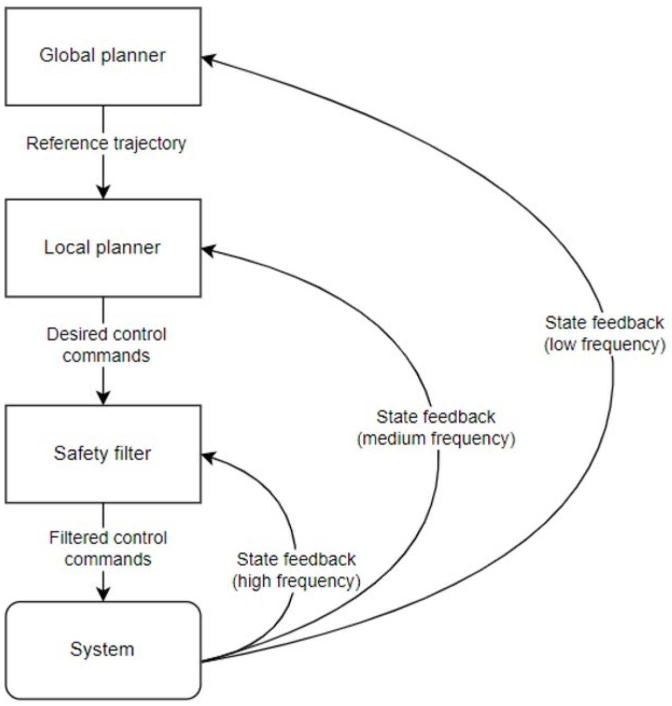

Global planner
Reference trajectory
Local planner
Desired control commands
Safety filter
Filtered control commands
System
State feedback (low frequency)
State feedback (medium frequency)
State feedback (high frequency)

Figure 3-1: Proposed hierarchical trajectory planner

3-2 Local planner

This sequence of configurations found by the first layer is used by a reference trajectory for the middle layer. The role of the middle layer is to track the global reference as closely as possible to ensure successful completion of the task, while accounting for any local short-term constraints. In other words, it must be possible to explicitly account for obstacles in the immediately reachable state space of the system (i.e., the portion of the state space that can be reached in a finite time horizon). This naturally leads us to using some form of Model Predictive Control (MPC), as these controllers are capable of making the system operate at its limit while avoiding constraint violations over the prediction horizon. The MPC would be formulated as such by adding the constraint that the EEFs should be aligned at the time samples. This would be a nonlinear equality constraint, which would naturally be non-convex. In order to ensure that these constraints are satisfied, we can make use of an interior point solvers which are known to be effective at ensuring that the found solution lies in the feasible set. Alternatively, one could make use of linear approximations as mentioned in Section 2-3-2.

The Nonlinear Model Predictive Control (NMPC) problem is implemented on a discrete time system, since the simpler design makes it faster to run on the hardware. However, this could potentially cause problems since the inter-sample behaviour is not accounted for. It is possible that the inter-sample behavior of the actual continuous time system will not violate the desired safety constraints, but we cannot be certain. In order to mitigate this uncertainty, we make use of a CBF based high-frequency controller.

3-3 Safety filter

The primary role of the lowest layer of the trajectory planner would be to minimize constraint violations as much as possible. Given that the desired safety constraint can be written as a CBF, it would then be possible to construct a controller that ensures that the condition in Definition 2.3 is satisfied.

The first step is to find a continuously differentiable function that can represent the rotation and translation constraints between the EEFs. We will start by defining the translation constraints, since they are easier to visualize. We want to ensure that the coordinates of the origin of the frame attached to EEF2 in the reference frame attached to the origin of EEF1 as a function of the joint variables (henceforth referred to as ${}^1t_2(q)$) stays within an axis-aligned (with respect to frame of EEF1) cube of the desired coordinates (henceforth referred to as ${}^1t_2^d$). Specifically, we desire that the error in each coordinate does not exceed some bound ϵ (which is taken to be the same for all the axes for simplicity, but can be different based on the use case), as visualized in Figure 3-2a. Mathematically, we can write this as

$$\begin{aligned}({}^1t_{2,x}(q) -{}^1 t_{2,x}^d)^2 \leq \epsilon^2 \\ ({}^1t_{2,y}(q) -{}^1 t_{2,y}^d)^2 \leq \epsilon^2 \\ ({}^1t_{2,z}(q) -{}^1 t_{2,z}^d)^2 \leq \epsilon^2\end{aligned} \tag{3-1}$$

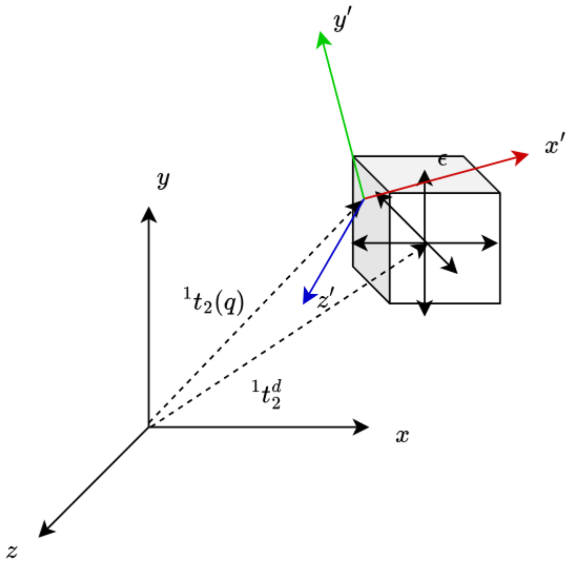
where ${}^1t_{2,i}(q)$ and ${}^1t_{2,i}^d$ refer to the i coordinate value of ${}^1t_2(q)$ and ${}^1t_2^d$ respectively. Equation 3-1 can then be used to write the translation CBFs as seen in Equation 3-2

$$\begin{aligned}h_1(q) &:= 1 - (({}^1t_{2,x}(q) -{}^1 t_{2,x}^d)/\epsilon)^2 &\geq 0 \\ h_2(q) &:= 1 - (({}^1t_{2,y}(q) -{}^1 t_{2,y}^d)/\epsilon)^2 &\geq 0 \\ h_3(q) &:= 1 - (({}^1t_{2,z}(q) -{}^1 t_{2,z}^d)/\epsilon)^2 &\geq 0\end{aligned} \tag{3-2}$$

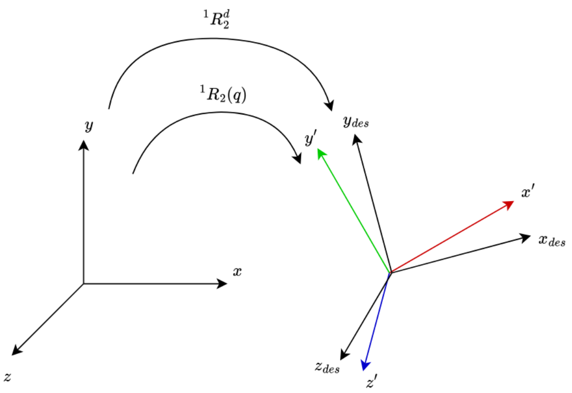
These equations can then be used to formulate the constraints on the CBF-Quadratic Programming (QP) controller

$$\begin{aligned}\frac{\partial h_1(q)}{\partial q}\dot{q} \geq -\alpha_s h_1(q) \\ \frac{\partial h_2(q)}{\partial q}\dot{q} \geq -\alpha_s h_2(q) \\ \frac{\partial h_3(q)}{\partial q}\dot{q} \geq -\alpha_s h_3(q)\end{aligned} \tag{3-3}$$

The rotation constraints can be constructed by exploiting the fact that rotation matrices are orthagonal. For example, assume we have rotation matrix from the reference frame attached to EEF2 to the reference frame attached to EEF1 as a function of the joint variables q (henceforth referred to as ${}^1R_2(q)$) and the desired rotation matrix from the reference frame attached to EEF2 to the reference frame attached to EEF1 (henceforth referred to as ${}^1R_2^d$). This is visualized in Figure 3-2b, where the desired orientation of the reference frame is coloured black, and the actual reference frame is coloured. For simplicity, let us refer to the actual frame as 2'.



(a) Translation error in frame transforms



(b) Rotation error in frame transforms

Figure 3-2: A visual interpretation of frame transformation errors

We can write a closed form expression of the transformation from the desired frame of EEF2 to the actual frame EEF2'

$$\begin{aligned} {}^2R_1(q)\ {}^1R_{2'} &= {}^2R_{2'} \\ \Rightarrow ({}^1R_2(q))^T\ {}^1R^d_{2'} &= {}^2R_{2'} \end{aligned} \tag{3-4}$$

In the case where the frame EEF2 exactly overlaps with frame EEF2' as desired, the axis will have a cosine similarity of 1, i.e., $x' \cdot x_{des} = y' \cdot y_{des} = z' \cdot z_{des} = 1$. We can then rewrite Equation 3-4 as

$$({}^1R_2(q))^T\ {}^1R^d_{2'} = I_3 = ({}^1R^d_{2'})^T\ {}^1R_2(q)$$

where I_3 refers to the 3×3 identity matrix. We can therefore use Equation 3-4 to obtain three bounds to constrain the rotation error. By ensuring that the diagonal elements of ${}^2R_{2'}$ are equal to 1, we can ensure that the EEFs are oriented as desired with respect to each other. We can write these closed form bounds as seen in Equation 3-5.

$$\begin{aligned} r_{11}(q)r^d_{11} + r_{12}(q)r^d_{12} + r_{13}(q)r^d_{13} \geq 1 - \varepsilon \\ r_{21}(q)r^d_{21} + r_{22}(q)r^d_{22} + r_{23}(q)r^d_{23} \geq 1 - \varepsilon \\ r_{31}(q)r^d_{31} + r_{32}(q)r^d_{32} + r_{33}(q)r^d_{33} \geq 1 - \varepsilon \end{aligned} \tag{3-5}$$

where $r_{ij}(q)$ and r^d_{ij} are the elements of the i^{th} row and the j^{th} column of ${}^1R_2(q)$ and ${}^1R^d_2$ respectively. ϵ refers to the maximum error we can tolerate in the dot product of the corresponding axes in frames EEF2 and EEF2', and as in the case of the translation error, can be different for each pair of axes based on the use case. We can interpret this as an upper bound on the angles between the corresponding axes, as seen in Equation 3-6

$$
\begin{aligned}
\theta_x &:= cos^{-1}(r_{11}(q)r^d_{11} + r_{12}(q)r^d_{12} + r_{13}(q)r^d_{13}) \leq cos^{-1}(1-\varepsilon) \\
\theta_y &:= cos^{-1}(r_{21}(q)r^d_{21} + r_{22}(q)r^d_{22} + r_{23}(q)r^d_{23}) \leq cos^{-1}(1-\varepsilon) \\
\theta_z &:= cos^{-1}(r_{31}(q)r^d_{31} + r_{32}(q)r^d_{32} + r_{33}(q)r^d_{33}) \leq cos^{-1}(1-\varepsilon)
\end{aligned} \tag{3-6}
$$

Note that only the lower bound needs to be explicitly defined, due to the fact that no element of a rotation matrix can exceed 1. Equation 3-5 can be reformulated into CBFs as seen in Equation 3-7

$$
\begin{aligned}
h_4(q) &:= 1 - \left(\frac{r_{11}(q)r^d_{11} + r_{12}(q)r^d_{12} + r_{13}(q)r^d_{13} - 1}{\epsilon}\right)^2 \geq 0 \\
h_5(q) &:= 1 - \left(\frac{r_{21}(q)r^d_{21} + r_{22}(q)r^d_{22} + r_{23}(q)r^d_{23} - 1}{\epsilon}\right)^2 \geq 0 \\
h_6(q) &:= 1 - \left(\frac{r_{31}(q)r^d_{31} + r_{32}(q)r^d_{32} + r_{33}(q)r^d_{33} - 1}{\epsilon}\right)^2 \geq 0
\end{aligned} \tag{3-7}
$$

The rate of change of these CBFs can then be bounded using Nagumo's theorem as seen before in Equation 3-3. In a similar manner, CBFs can be constructed for obstacle avoidance by ensuring that the distance from an obstacle is greater that 0 as done in [34, Equation 6] and shown more concretely later in Equation 4-4.

The resulting controller would be of the form as seen in Problem 3-8, with $h(\cdot)$ referring to a vector of all the control barrier functions discussed so far.

$$
\begin{aligned}
\min_{u} \quad & ||u - u_{des}||^2 \\
s.t. \quad & \frac{\partial h(x)}{\partial x} f(x) + \frac{\partial h(x)}{\partial x} g(x)u \geq -\alpha(h(x)) \\
& A_{\mathcal{U}} u \leq b_{\mathcal{U}}
\end{aligned} \tag{3-8}
$$

Note how there is no focus on task execution (from Requirement 3) in this controller. This is because the role of ensuring successful task execution is delegated entirely to the global planner and the NMPC in the top and the middle layer, with the resulting information being represented in the NMPC output u_{des}. The sole role of the CBF based controller is to ensure that safety constraint violations are kept to a minimum. By correctly designing the safe set that the CBF-QP controller attempts to render invariant, it is possible to ensure that our actual state safety constraints are not violated. This ensures the satisfaction of Requirement 1 and partially satisfies Requirement 2.

In the experiments shown in this work, the system is modelled as being velocity-controlled (as shown in Chapter 4), which is a system with a relative degree of 1. However, this design can easily be extended to cases where the robot is controlled using joint accelerations or torques (systems with a relative degree of 2 or more), by making use of Exponential Control Barrier Function (ECBF)s as shown in [32].

It may occur that Requirements 1 and 2 may conflict when the system is very close to a collision, rendering the CBF-QP infeasible. In that scenario, it is proposed to use a fall-back control law K_{fb} to trigger an emergency stop (which is a common feature in industrial manipulators as pointed out in [34]), before defaulting to the CBF-QP. This ensures that Requirement 1 always takes precedence, thus ensuring full satisfaction of Requirement 2.

Chapter 4

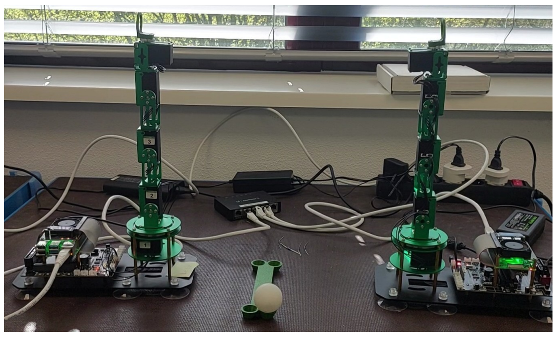
Case study: dual-arm pick and place

This chapter details simulations that were conducted on a test setup to demonstrate the proposed motion planning framework in action. The test setup is described in the following section along with the corresponding kinematic model. This information is then used to construct the mid-level Model Predictive Control (MPC) and low-level Control Barrier Function (CBF) controllers. The proposed planner is then run in three scenarios: one experiment without an obstacle, one with the presence of a non-occluding obstacle and a third where the obstacle occludes the global plan. These runs are then compared in terms of time taken for execution, as well as the maximum violation of the constraints.

4-1 Setup description and modelling

In order to test the effectiveness of the proposed trajectory planner, it was tested on a test setup of two Yahboom DOFBOT arms (shown in Figure 4-1a), spaced 15 cm apart and facing each other, having to pick up a ping-pong ball from a specified pick position and carrying it to a specified place position. The desired task (as stated in Requirement 3), is to reach the place position as quickly as possible without dropping the ball. In order to avoid dropping the ball, the arms must keep a constant distance of 4 cm between them (the diameter of the ball), with an error tolerance of 3mm. The rings on the robot (as seen in Figure 4-1b) must always have their centers aligned. Violating either of these conditions will lead to the ball being dropped. The specification sheet of the arms has been provided in Apendix B.

The arm itself may be modelled as a kinematic chain, with each joint contributing one variable. A point to note is that the arm has 5 movable joints, meaning it has 5 Degrees of Freedom (DOF). This could pose issues since most readily available motion planners are designed to work with robots with 6 or more DOF. In order to get around this, we model a 6^{th} dummy joint at the center of the ring since the task is invariant to the actions of such a revolute joint. It is then possible to write the transformation matrices to transform the frame of reference from joint i to joint $i+1$, as seen in Table B-1.


(a) Setup with the Yahboom DOFBOT robots

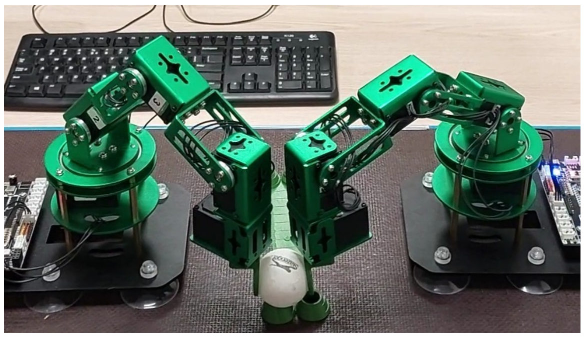

(b) Robots in the pick-up configuration

Figure 4-1: Physical setup with the robots

The arms are operated using servos that take joint positions as inputs. In order to make this amenable to planner proposed in Chapter 3, we have chosen to model the system as being velocity controlled as seen in Equation 4-1, where $q(t)$ represents the joint angles (the states) and $\omega(t)$ represents the joint velocities (the control inputs). The setup is simulated in Robot Operating System (ROS)2. The setup is first described using a Unified Robot Description Format (URDF) file and then integrated with MoveIt2 using the setup assistant. This integration also creates a trivial simulation of the system (also known as a digital twin) which is used to simulate the effect of velocity commands on the robots. A schematic representation of the simulation can be seen in Figure 4-2, which shows how the different programs that make up the simulation are connected. The MoveIt2 simulation integrates the velocity commands that are output by the CBF-Quadratic Programming (QP) controller using Heun's method, to generate a position and velocity profile for the simulated robot that is visualized in RViz.

$$\dot{q}(t) = \omega(t) \tag{4-1}$$

Two key assumptions are made in this experiment:

Assumption 1: The robots are moving slow enough that the effect of their inertial dynamics can be ignored.

Assumption 2: The inbuilt control loop of the robot is capable of accurately tracking the desired trajectory profile generated by the proposed planner.

In order to ensure that these assumption remain reasonable, we limit the joint velocities to be well within what the servos are capable of following. As can be seen from the specification sheet in Appendix B, the servos are capable of rotating at a maximum speed of 200 degrees per second (3.5 rad/s). We limit the maximum velocity that can be commanded by the planner to be within a magnitude of 0.49 rad/s, less than 15% of the maximum velocity that the servo is capable of reaching.

4-1-1 Global Planner: RRT-Connect

The first point of order is to construct the global planner which will generate the reference joint trajectories that the controllers at the lower layers should follow. In this experiment,

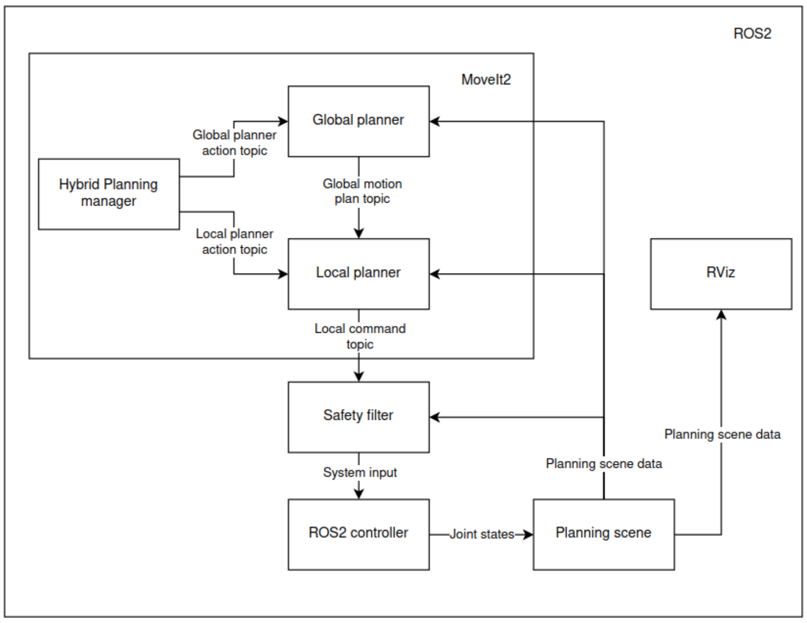


Figure 4-2: Simulation setup

we implement the global planner to work in an event-triggered manner, with the planner only running when requested for a motion plan. The motion planning request can either be triggered by the user to generate a path to a user-specified point, or by the local planner to request a new reference in case it is stuck at a local minima and unable to make progress.

The global planner makes use of MoveIt2's inbuilt planning planning pipeline, which in turn can make use of any of the motion planning algorithms available in Open Motion Planning Library (OMPL) [40]. In this particular implementation, we use Rapidly exploring Random Trees (RRT)-Connect [41], simply because it is Moveit2's default motion planner and returns the first motion plan that is found. However, in practice, it can be replaced with any of the other planners available in OMPL (this was confirmed by testing the code by making use of SPArse Roadmap Spanner (SPARS) instead of RRT-Connect). Since MoveIt2 separates the components of OMPL from the user, it was not possible to design custom state samplers to exploit heuristics of the problem, leading to very long planning times. To circumvent this, the planning pipeline was given tolerances of approximately 15 cm and a cosine similarity error tolerance of approximately 0.03. Once a plan was found, the waypoints were projected into the free valid configuration space using a custom NLP as seen in Problem 4-2, and the resulting sequence of waypoints is treated as the global reference trajectory.

$$\begin{aligned} &\min_x (x - x_{des})^T (x - x_{des}) \\ s.t. \quad & {}^1t_2(x) = \; {}^1t_2 \\ & {}^1R_2(x)^2R_1 = \; I_{3\times 3} \\ & ({}^wt_1(x) - x_{obs})^2 >= r_{obs}^2 \\ & ({}^wt_2(x) - x_{obs})^2 >= r_{obs}^2 \end{aligned} \tag{4-2}$$

When the global planner successfully computes a trajectory, it sends the entire trajectory to the local planner, where it is stored and used as a reference. The global planner then becomes inactive until it is queried again for a new motion plan.

4-1-2 Local planner: NMPC

This section will deal with how middle layer of the planner is designed. The middle layer is constructed of two parts: a trajectory operator (akin to a target selector in output MPC) and a local constraint solver (which is the NMPC itself). The trajectory operator receives the entire sequence of waypoints and forwards the initial waypoint to the local constraint solver. Both the trajectory operator and the local constraint solver have access to the planning scene. When the trajectory operator detects that the system is sufficiently close to the current waypoint, it updates the reference to the next waypoint and forwards the new reference to the local constraint solver. This can also happen in case the trajectory operator determines that the waypoint has been occluded by an obstacle. In this example, we assume that the obstacle (if it exists) moves only once, after the first global plan has been created. Therefore, if the trajectory operator detects that the waypoint is occluded, it immediately discards it and moves on to the next waypoint. By design, it is assumed that the goal state is never occluded.

For this case study, a simple representation of the alignment constraints are used. Note that when there is a constant frame transformation matrix between the two End Effector (EEF)s, that implies that each EEF is holding a constant pose in the frame of the other. In other words, EEF2 is always at a constant position and holding a constant orientation in a reference frame attached to EEF1 and vice versa. One can express this condition as seen in Equation 4-3.

$$\begin{aligned} {}^1t_2(x) &= {}^1 t_2^d \\ {}^2t_1(x) &= {}^2 t_1^d \end{aligned} \tag{4-3}$$

${}^it_j(x)$ refers to the position of EEFj in the reference frame of EEFi as a function of the joint values. ${}^it_j^d$ refers to the constant desired position of EEFj in the reference frame of EEFi.

The obstacle avoidance constraints is written as a constraint that the position of the EEF in the world frame lies in the exterior of the (assumed to be spherical) obstacle, as seen in Equation 4-4.

$$||{}^wt_i(x) -{}^w t_{oj}||^2 > r_{oj}^2 \tag{4-4}$$

Similar to before, ${}^wt_i(x)$ and ${}^wt_{oj}$ refers to the position of EEFi and obstacle j in the world frame, respectively and r_{oj} is the radius of the j^{th} obstacle.

The local constraint solver attempts to track the reference using Nonlinear Model Predictive Control (NMPC). The NMPC controller is written using IMPACT [42], a toolchain allowing for rapidly prototyping NMPC controllers as well as code-generating C++ artifacts for deployment. IPOPT [40] is used as a solver due to interior point methods being very successful at ensuring that the problem constraints are met. In the NMPC structure, we use integrator dynamics along with the alignment and obstacle-avoidance constraints, as seen in Problem 4-5. The prediction horizon N is set to 5 and the value of β is set to 5 as well. The choice of weighting the terminal cost for stability was made since constructing the terminal set was non-trivial.

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} x(k)^T x(k) + \beta x(N)^T x(N) & \\
s.t. \quad & x(k+1) = x(k) + \Delta_T u(k) & \forall k \in \mathbb{Z}_{0:N-1} \\
& x_{min} \leq x(k) \leq x_{max} & \forall k \in \mathbb{Z}_{1:N} \\
& u_{min} \leq u(k) \leq u_{max} & \forall k \in \mathbb{Z}_{0:N-1} \\
& {}^1t_2(x(k)) = {}^1t_2^d & \forall k \in \mathbb{Z}_{1:N} \\
& {}^2t_1(x(k)) = {}^2t_1^d & \forall k \in \mathbb{Z}_{1:N} \\
& ||{}^wt_1(x(k)) - {}^wt_o||^2 \geq r_o^2 + \epsilon_{safety} & \forall k \in \mathbb{Z}_{1:N} \\
& ||{}^wt_2(x(k)) - {}^wt_o||^2 \geq r_o^2 + \epsilon_{safety} & \forall k \in \mathbb{Z}_{1:N} \\
& x(0) = x_0 &
\end{aligned} \tag{4-5}$$

The NMPC was exported as a C++ function that was used to write a local constraint solver plugin for the local planner in MoveIt2's hybrid planning framework. The local planner node was run at a frequency of 10 Hz. At each control interval, the solution is warmstarted using the solution from the previous control interval.

4-1-3 Safety filter: CBF-QP

The lowest layer is constructed using a CBF-QP controller. The controller accepts only the desired velocity commands as an input and projects the velocity command into the space defined by the CBF and the control input limits. It is given that we are allowed an error of 3 mm in the distance between the arms. In order to avoid violating that limit, we design the controller such that it attempts to keep the error within a more conservative bound of 0.1 mm. This ensures that even if the system violates our designed bound, we will not violate the actual safety limits of our system. We also limit the rotation error by ensuring the axes pairs have a cosine similarity (or dot product) of 0.9999 or higher. In geometric terms, we want the angle between each axis pair to be $\leq\sim 0.014$ radians.

This results in us setting the values of ϵ and ε (from Equations 3-2 and 3-7 respectively) to a value of 10^{-4}. The CBFs $h_1(\cdot)$ to $h_6(\cdot)$ are as defined in Section 3-3. Equation 4-4 was used as collision avoidance CBFs $h_7(\cdot)$ and $h_8(\cdot)$ for EEF1 and EEF2 respectively. The radius of the obstacle in the environment (when it exists) was inflated by 1.5 cm in the CBF, so as to ensure that the EEF maintained some distance from the obstacle. The values of α from Equation 3-3 was set to 7.9 for CBFs $h_1(\cdot)$ to $h_6(\cdot)$ and to 10.0 for CBFs $h_7(\cdot)$ and $h_8(\cdot)$.

The safety filter was written using CasADi [43], with OSQP [44] used as the backed solver. The CasADi program was written in Python, with the code-generation feature being used to generate C++ artifacts that could then be used with a ROS2 node. The safety filter was run as a standalone node that was run at a 100 Hz.

4-2 Experiments

In order to test the effectiveness of the proposed planner, three sets of experiments were carried out. In order to compare the performance of the proposed controller design, the reference generated was tracked using two different controllers: a discrete-time NMPC, and a discrete-time NMPC cascaded with a high frequency safety filter. To reduce the variance that occurred from the randomness in the global planner, a previously successful global plan was stored and used as the reference for all the experiments (visualized in Figure 4-3). The system was simulated using ROS2 and RViz, meaning the system still evolved while the solvers were running iterations, or nodes were communicating. This led to non-deterministic (albeit minute) delays which caused some variance. In order to account for this variance, the data was tabulated across 10 runs for each experiment, with the average value written in the table along with the standard deviation.

Pick and place without obstacle In the first set of experiments, the planner had to move from the pick state to the place state in the absence of obstacles. The statistics for the translation errors, the rotation errors and the time duration the conditions are violated are tabulated in Tables 4-1, 4-2 and 4-3 respectively.

Pick and place with obstacle In the second set of experiments, the planner had to move from the pick state to the place state while avoiding an obstacle that was present in the environment. The statistics for the translation errors, the rotation errors and the time duration the conditions are violated are tabulated in Tables 4-4, 4-5 and 4-6 respectively.

Pick and place with occluded waypoints In the third set of experiments, the planner had to move from the pick state to the place state while avoiding an obstacle in the environment. After the initial global plan was computed, the obstacle changes its location, causing some waypoints to become invalid and getting dropped from the global plan. The statistics for the translation errors, the rotation errors and the time duration the conditions are violated are tabulated in Tables 4-7, 4-8 and 4-9 respectively.

4-3 Results

4-3-1 Pick and place without obstacle

In this experiment, the arms had to move from positions of $(0.075, -0.034, 0.029)^T$ and $(0.075, -0.030, 0.029)^T$ to positions of $(-0.075, -0.047, 0.028)^T$ and $(-0.075, -0.043, 0.028)^T$ for arm 1 and arm 2 respectively. They had to complete this motion while maintaining a constant relative orientation of $(0, 0, -1, 0)^T$ between them (represented in standard quaternion notation of $(w, x, y, z)^T$). The simulations are visualized in Figures 4-4a and 4-4b.

In order to visualize the effect of the CBF-QP safety filter, the inputs generated by the NMPC are visualized in Figures 4-5a and 4-5b while the actual inputs applied to the system are visualized i Figures 4-6a and 4-6b respectively. Figures 4-7a and 4-7b provide an enhanced

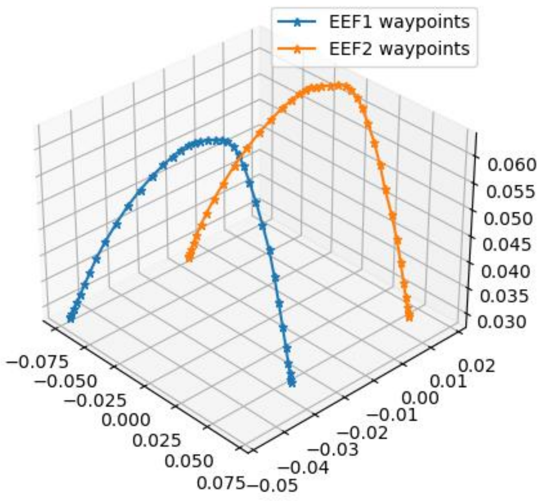
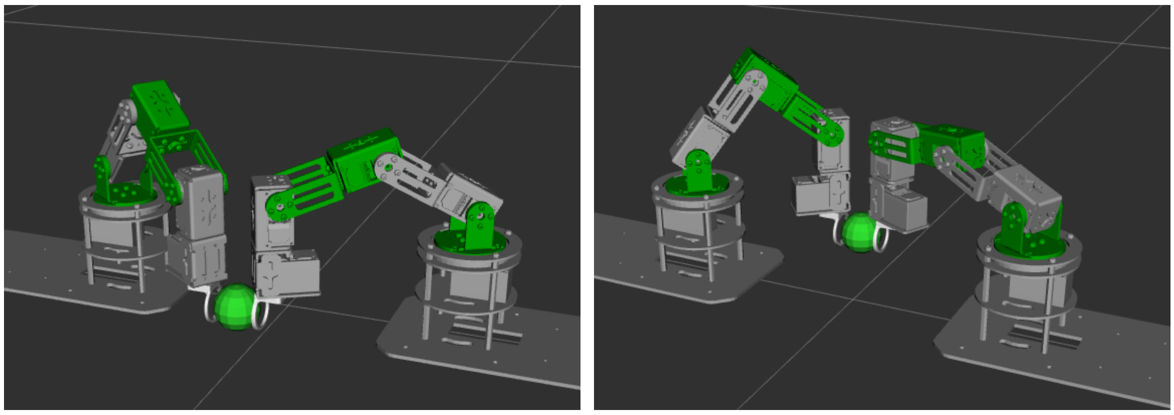


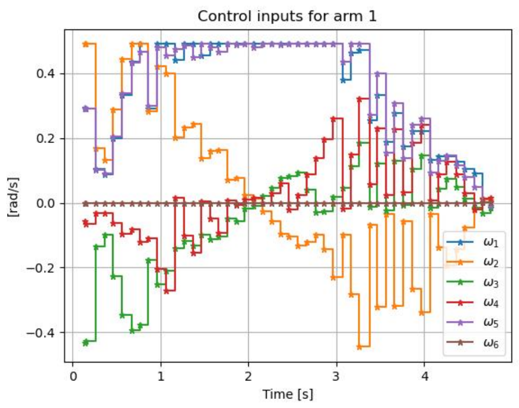
Figure 4-3: The global reference generated by the global planner



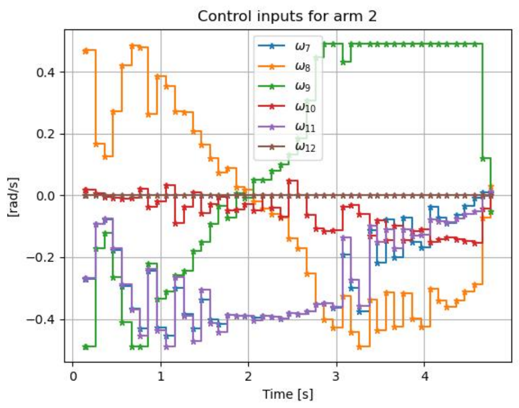
(a) Robots in the pick configuration

(b) Robots in the place configuration

Figure 4-4: A visualization of the pick and place simulation in the absence of obstacles.

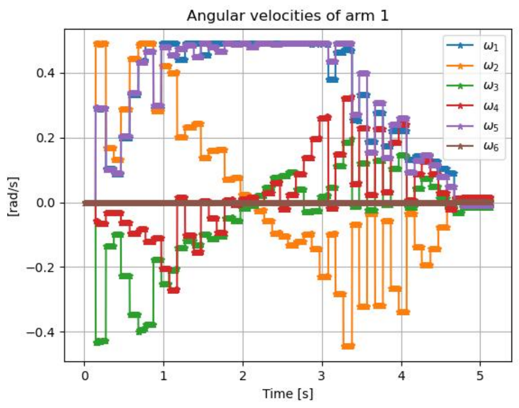


(a) NMPC control inputs for arm 1

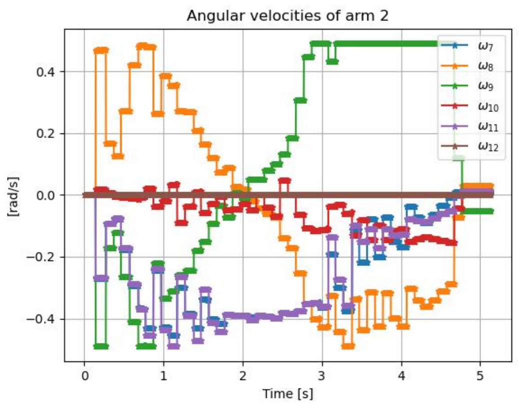


(b) NMPC control inputs for arm 2

Figure 4-5: Control actions determined by the NMPC when there are no obstacles.

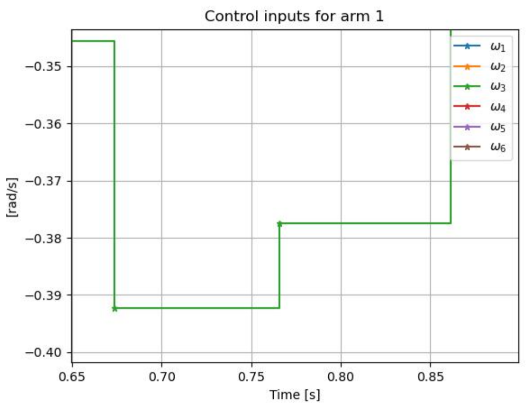


(a) Control inputs for arm 1

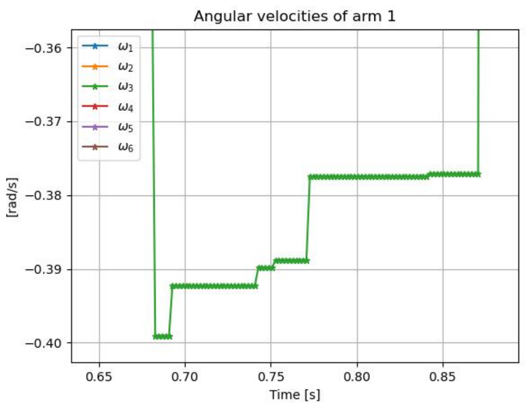


(b) Control inputs for arm 2

Figure 4-6: Actual inputs applied to the system after filtering the NMPC outputs from Figures 4-5a and 4-5b through the CBF-QP.



(a) The input determined by the NMPC



(b) The actual input applied to the system

Figure 4-7: An example of the effect of the safety filter, where it changes the actual input to the system to satisfy the barrier conditions

view of one of the inputs to arm 1. It is seen that the CBF-QP adds its own control action to the nominal control input generated by the NMPC.

It can be observed from Table 4-1, that the addition of a CBF-QP filter was capable of reducing the average maximum error recorded in the z direction to about 50% of the value observed when the discrete time NMPC is used stand-alone. This can be visually observed by comparing Figures 4-8a and 4-9a. This is highlighted even more in Table 4-3. Predictably, the combination of NMPC with CBF-QP is slower by $\sim 1s$ than the NMPC standalone, since the safety filter may reduce the speed to avoid crossing the error threshold. However, the presence of the safety filter also reduces the average time during the run where the error crosses the threshold, reducing it from $\sim 1s$ to $\sim 0.24s$.

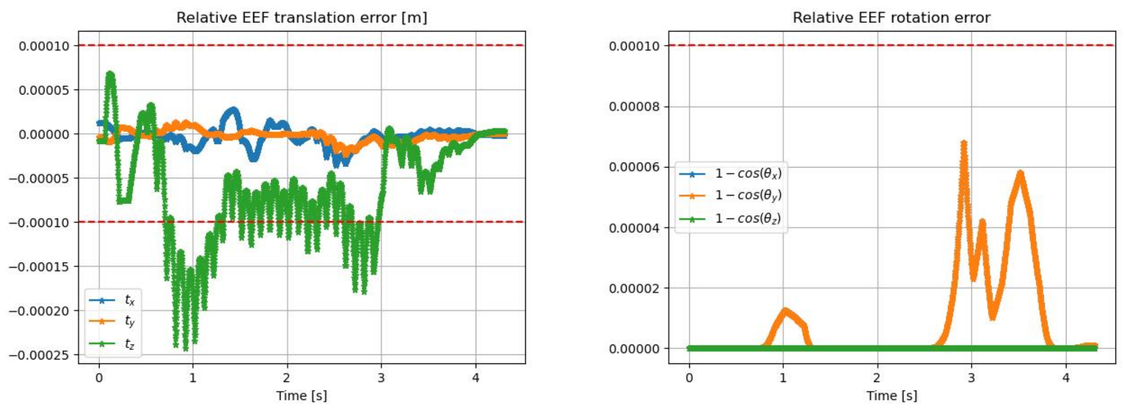
4-3-2 Pick and place with obstacle

The second experiment models the same pick and place scenario as before, in the presence of a spherical obstacle of radius 2.5 cm with its center at the origin. The arms have to perform the same motion as before while ensuring that there is no collision with the obstacle. The simulations are visualized in Figures 4-10a and 4-10b.

As before, it is observed from Table 4-6 that the NMPC alone is faster in completing the task when compared to the NMPC in cascade with a CBF-QP. However, the addition of the CBF-QP does lead to lower maximum violations as evidenced by Tables 4-4 an d 4-5, as well

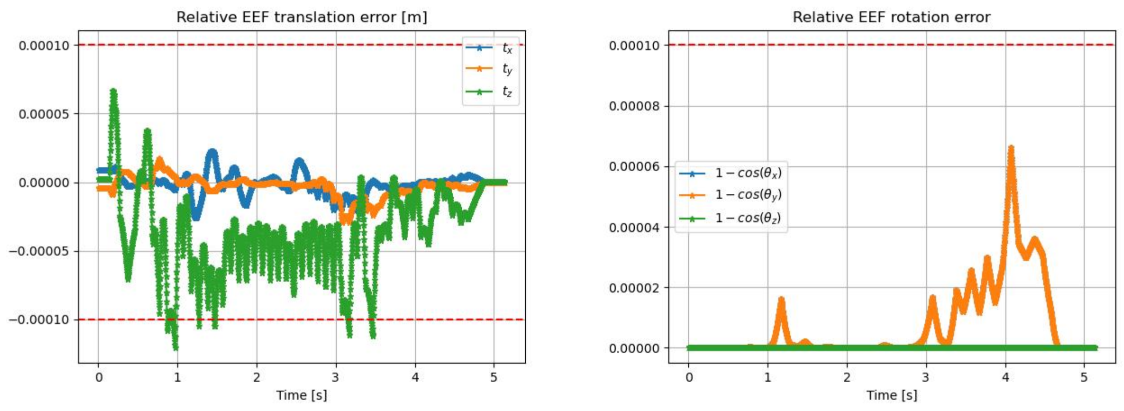
Max translation error (10^{-4} m)	**Controller type**	
	NMPC	**NMPC + CBF-QP**
$\mathbf{t_x}$	0.378 ± 0.040	0.300 ± 0.043
$\mathbf{t_y}$	0.248 ± 0.047	0.345 ± 0.063
$\mathbf{t_z}$	2.602 ± 0.422	1.480 ± 0.347

Table 4-1: Translation errors recorded in in a pick and place task without obstacles.



(a) Translation errors **(b)** Rotation errors

Figure 4-8: Measured alignment errors using NMPC alone in the absence of obstacles.



(a) Translation errors **(b)** Rotation errors

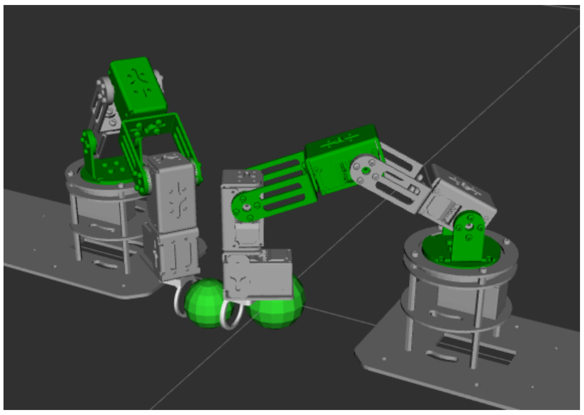
Figure 4-9: Measured alignment errors when using NMPC with a CBF-QP in the absence of obstacles.

Max rotation error (rad)	**Controller type**	
	NMPC	**NMPC + CBF-QP**
$\theta_\mathbf{x}$	0.012 ± 0.000	0.012 ± 0.000
$\theta_\mathbf{y}$	0.012 ± 0.000	0.012 ± 0.000
$\theta_\mathbf{z}$	0.000 ± 0.000	0.001 ± 0.000

Table 4-2: Rotation errors recorded in a pick and place task without obstacles.

Duration (s)	Controller type	
	NMPC	NMPC + CBF-QP
Execution	4.009 ± 0.002	4.856 ± 0.037
$h_1(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_2(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_3(x) < 0$	1.087 ± 0.096	0.196 ± 0.087
$h_4(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_5(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_6(x) < 0$	0.000 ± 0.000	0.000 ± 0.000

Table 4-3: Execution time and duration of CBF violations recorded in a pick and place task without obstacles.


(a) Robots in the pick configuration

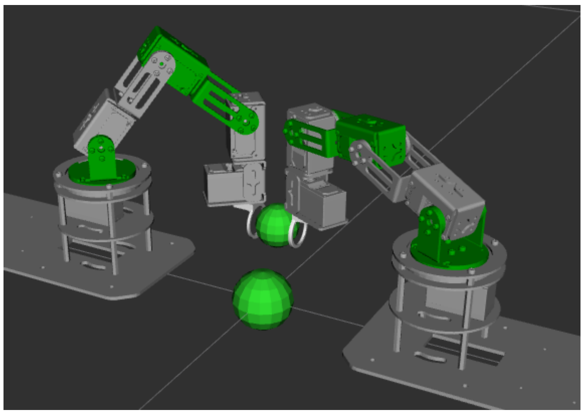
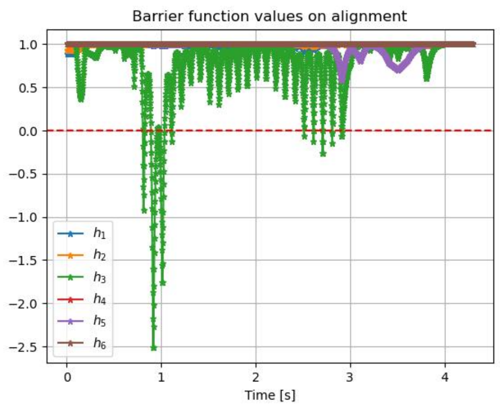
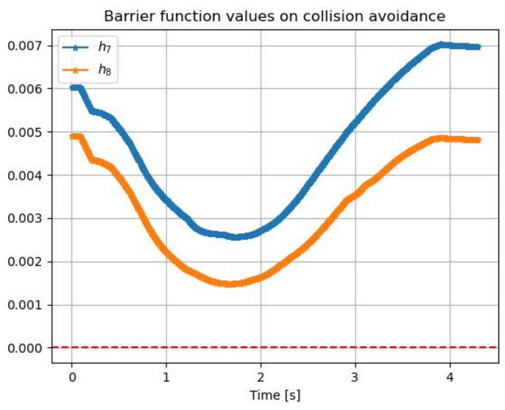

(b) Robots in the place configuration

Figure 4-10: A visualization of the pick and place simulation in the presence of an obstacle.

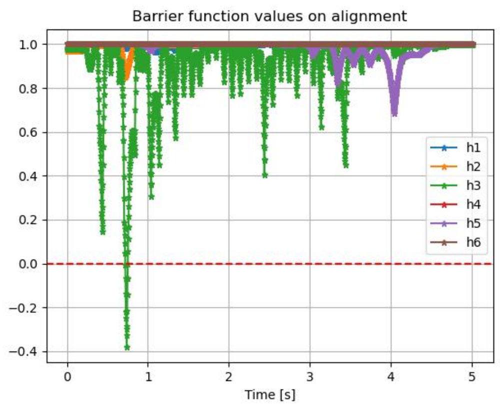


(a) The CBF values for alignment

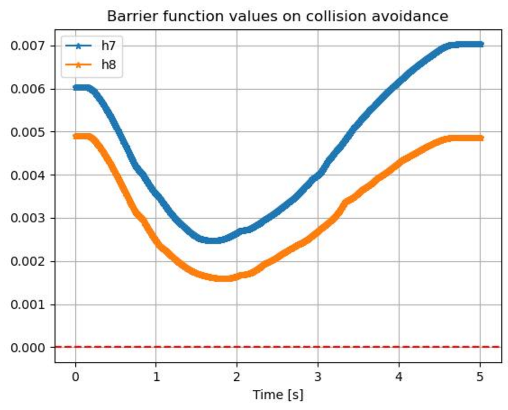


(b) The CBF values for collision

Figure 4-11: CBF values with NMPC for a pick and place task in the presence of an obstacle.



(a) The CBF values for alignment



(b) The CBF values for collision

Figure 4-12: CBF values with NMPC and CBF-QP for a pick and place task in the presence of an obstacle.

as lower violation times (if any) as seen from Table 4-6. This is visually seen in Figures 4-11a and 4-12a. As seen from comparing Figures 4-11b and 4-12b, both controllers are adept at avoiding collisions.

4-3-3 Pick and place with occluded waypoints

In the third experiment, the obstacle was moved after the initial global plan was created, with its center at $[0, 0, 0.025]^T$. This was done to simulate the possibility that the global reference plan becomes faulty due to the environment changing, and to observe whether the NMPC (both with and without the CBF-QP) is capable of reaching the desired place position. The global plan was purposely selected such that some waypoints be occluded when the obstacle moved (due to arm 2 colliding with the obstacle), as seen from Figures 4-17. In the implementation, any such occluded waypoint was skipped during execution. The simulations are visualized in Figures 4-13a and 4-13b.

By comparing execution times from Tables 4-6 and 4-9, it was observed that the planner still managed to reach the place position in approximately the same time as before. with marginal increases in translation alignment errors, as can be seen from Tables 4-4 and 4-7. However, the reader will notice that there is a large spike with the rotation error when using only the NMPC, both from Table 4-8 and from Figure 4-15a. This occurs from the way the NMPC constraints are formulated in Equation 4-3: when the two EEF frames have a collinear set

Max translation error $(10^{-4}\ m)$	**Controller type**	
	NMPC	**NMPC + CBF-QP**
$\mathbf{t_x}$	0.286 ± 0.084	0.268 ± 0.120
$\mathbf{t_y}$	0.18 ± 0.042	0.288 ± 0.182
$\mathbf{t_z}$	2.017 ± 0.841	1.312 ± 0.340

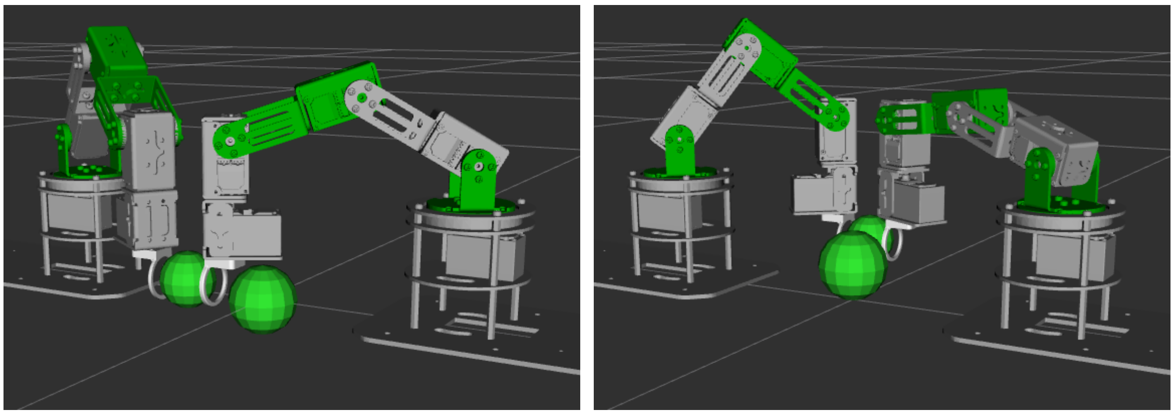
Table 4-4: Translation errors recorded in pick and place tasks in the presence of an obstacle.

Max rotation error (rad)	**Controller type**	
	NMPC	**NMPC + CBF-QP**
$\theta_{\mathbf{x}}$	0.011 ± 0.000	0.011 ± 0.001
$\theta_{\mathbf{y}}$	0.011 ± 0.000	0.011 ± 0.001
$\theta_{\mathbf{z}}$	0.000 ± 0.000	0.001 ± 0.000

Table 4-5: Rotation errors recorded in pick and place tasks in the presence of an obstacle.

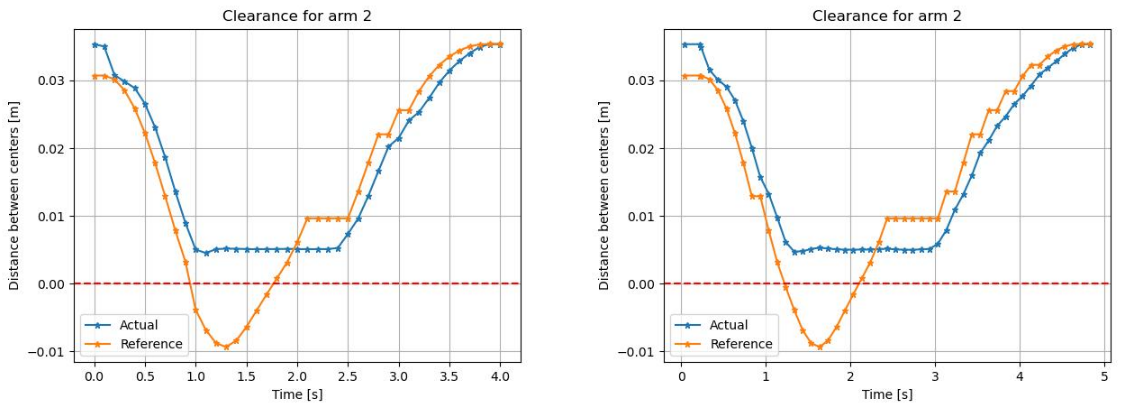
Duration (s)	**Controller type**	
	NMPC	**NMPC + CBF-QP**
Execution	4.025 ± 0.042	4.892 ± 0.067
$h_1(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_2(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_3(x) < 0$	0.204 ± 0.081	0.077 ± 0.085
$h_4(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_5(x) < 0$	0.000 ± 0.000	0.000 ± 0.000
$h_6(x) < 0$	0.000 ± 0.000	0.000 ± 0.000

Table 4-6: Execution time and duration of CBF violations recorded in pick and place tasks in the presence of an obstacle.



(a) Robots in the pick configuration **(b)** Robots in the place configuration

Figure 4-13: A visualization of the pick and place simulation in the presence of an obstacle. Note how the obstacle has moved in between these two instants. This is to ensure that it occludes some of the waypoints in the global plan.



(a) For NMPC **(b)** For NMPC with CBF-QP

Figure 4-14: The distance of the tracked waypoint versus the actual distance of the arm is visualized. When the clearance is negative, it denotes a collision (occlusion in the case of a waypoint).

of unit vectors, the rotation around those vectors becomes unconstrained. The NMPC was still formulated with these constraints, since it was observed that using a formulation similar to the CBF-QP to represent rotation caused the IPOPT solver to invariably throw an error saying that the solution quickly becomes infeasible. This effect is somewhat mitigated by the global reference, since the NMPC is always seeking to drive the state to a fixed configuration that does satisfy the alignment constraints.

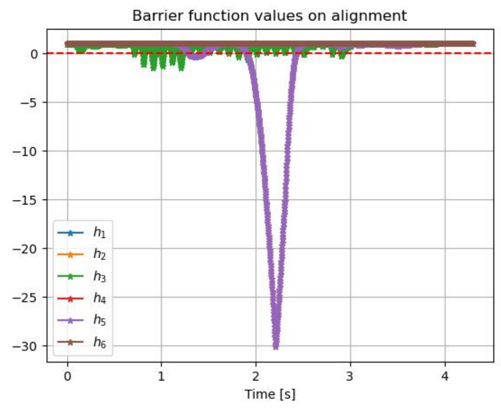
4-4 Discussion

In this section, we will cover some observations that were made and the resulting inferences that were made.

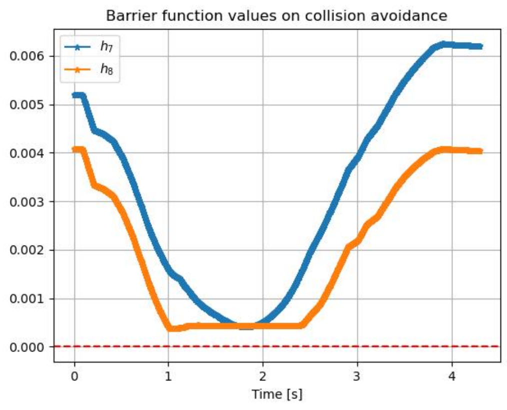
One conclusion that can be made from looking at the data is that the presence of a safety filter does indeed lead to an improvement in the alignment in the arms. As expected, it is not able to completely avoid violations due to unmodelled effects such as the delay in communication (or in the case of a real robot, servo accuracy and link deformation as well). However, it is able to mitigate their effects, as evidenced from the observation made in Section 4-3-3: though the NMPC alone did see a big spike in the rotation error, this effect was more muted when a CBF-QP was present. Furthermore, it can be observed that the actual alignment condition, which was to ensure the EEFs are never more than 3 mm apart, was not violated. This can

Max translation error $(10^{-4}\ m)$	**Controller type**	
	NMPC	**NMPC + CBF-QP**
$\mathbf{t_x}$	0.988 ± 0.136	1.148 ± 0.235
$\mathbf{t_y}$	0.353 ± 0.028	0.815 ± 0.094
$\mathbf{t_z}$	2.132 ± 1.047	1.529 ± 0.512

Table 4-7: Translation errors recorded in a pick and place task with occluded waypoints.

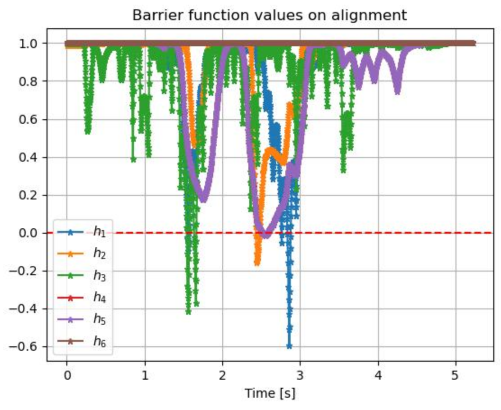


(a) The CBF values for alignment

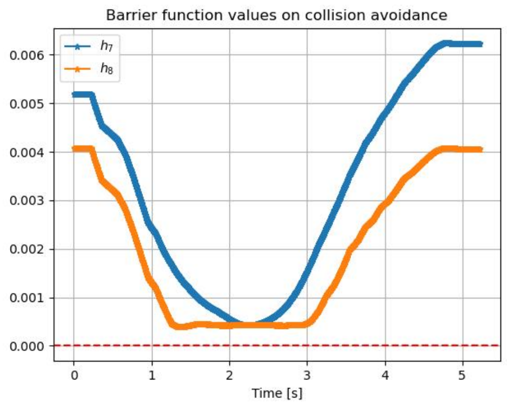


(b) The CBF values for collision

Figure 4-15: CBF values with NMPC for Experiment 3



(a) The CBF values for alignment



(b) The CBF values for collision

Figure 4-16: CBF values with NMPC and CBF-QP when a part of the global plan is occluded.

Max rotation error (rad)	**Controller type**	
	NMPC	**NMPC + CBF-QP**
$\theta_\mathbf{x}$	0.033 ± 0.000	0.014 ± 0.000
$\theta_\mathbf{y}$	0.033 ± 0.000	0.014 ± 0.000
$\theta_\mathbf{z}$	0.000 ± 0.000	0.001 ± 0.000

Table 4-8: Rotation errors recorded in a pick and place task with occluded waypoints.

Duration (s)	Controller type	
	NMPC	NMPC + CBF-QP
Execution	4.046 ± 0.055	4.868 ± 0.053
$h_1(x) < 0$	0.007 ± 0.013	0.054 ± 0.055
$h_2(x) < 0$	0.000 ± 0.000	0.004 ± 0.014
$h_3(x) < 0$	0.304 ± 0.065	0.11 ± 0.078
$h_4(x) < 0$	0.675 ± 0.019	0.040 ± 0.057
$h_5(x) < 0$	0.675 ± 0.019	0.040 ± 0.057
$h_6(x) < 0$	0.000 ± 0.000	0.000 ± 0.000

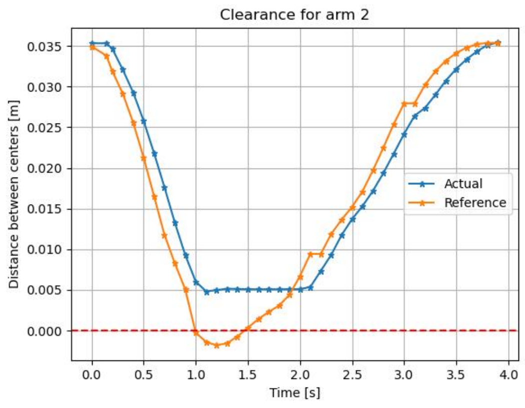
Table 4-9: Execution time and duration of CBF violations recorded in a pick and place task with occluded waypoints.

be attributed to Proposition 2.5, where by using a more conservative error bound we made the CBF more robust to such model errors. However, since there is no consideration for task performance, the trade-off in terms of execution time is not ideal. This could potentially be addressed by adding a Control Lyapunov Function (CLF) to the QP formulation in Problem 3-8, as done in [30].

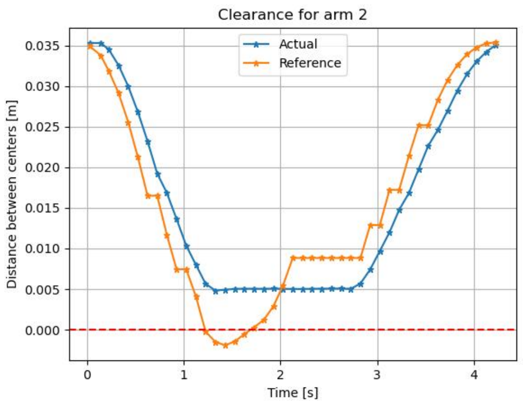
One interesting observation was made by comparing Tables 4-3 and 4-6 that there was in fact a reduction in the time that CBF $h_3(\cdot)$ (violation of translation constraint along the z-axis) in both cases compared to Experiment 1. Tables 4-1 and 4-4 show that there is a more muted effect on the maximum constraint violation observed as well. Given that the global reference is the same across experiments, and that it is observed both when NMPC is used standalone as well as when it is cascaded with the CBF-QP, it can be inferred that this behaviour is arising from the NMPC itself. It was confirmed through additional experiments that the NMPC did perform better in this scenario when it was formulated without collision avoidance constraints. It was noted in these experiments that IPOPT took fewer iterations to converge to a solution when the collision avoidance constraints were removed. This suggests that using a NMPC formulation without collision avoidance constraints, as opposed to treating the obstacle as being far away as in the current implementation, could lead to better alignment when there are no obstacles in the workspace.

In the context of the original problem of enabling a Computed Tomography (CT) scanning system, it is possible that obstacles may enter and leave the workspace (e.g. workers on an assembly line or medical staff in an operating theater). In such a scenario, using a switched controller in order to activate or deactivate a given number of collision avoidance constraints as necessary could help suppress the impact on alignment during scanning.

Another point to note is that the effect of the global planner was removed in the experiments since it was a source of randomness. However, it was observed during a number of experiments that the quality of the global plan could impact the observed misalignment. To give an example of this, we compare the visualized results from Experiment 3 with recorded data taken from a separate test with all the conditions being the same except that the global plan is created on demand. By comparing Figures 4-17, 4-17a and 4-17b, it is observed that not as many waypoints are occluded. By comparing the corresponding alignment errors from Figures 4-15a with 4-18a and 4-16a with 4-18b, it can be inferred that the global plan does impact the alignment errors observed.

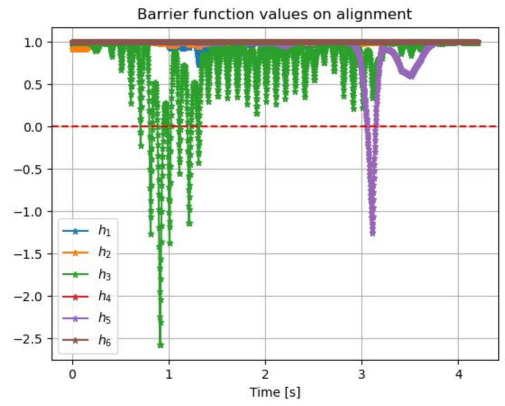


(a) For NMPC

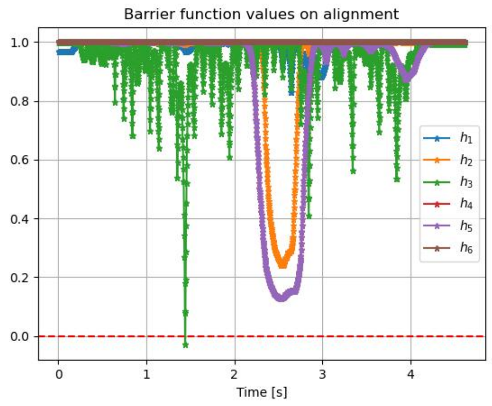


(b) For NMPC with CBF-QP

Figure 4-17: Distance of arm 2 from obstacle in the reference and actual trajectories



(a) When using NMPC



(b) When using NMPC and CBF-QP

Figure 4-18: CBF values for Experiment 3 when fewer waypoints get invalidated

As the global planner was not the main focus of this work, a readily available global planner in MoveIt2 was adopted. However, for the actual use case, it is paramount to build a proper implementation using the problem heuristics. This same topic is highlighted as future work in the next chapter.

Chapter 5

Conclusions and Future Work

This thesis has investigated possible methods to plan trajectories with very fine alignment requirements for dual robotic arm systems. Through a review of the existing literature, a hierarchical trajectory planning framework has been proposed that can satisfy the design requirements. The framework was designed trying to keep it as general as possible, so that it could be employed with multiple robot platforms with minimal alterations. Finally, a case study was performed for a pick and place task in order to test the if the architecture could satisfy the design requirements.

The overarching goal that inspired this thesis was to design a motion planning framework that can be used with dual-arm imaging systems such as X-ray and Computed Tomography (CT) imaging. While the case study confirmed how one could enable point-to-point motion wile keeping the arms aligned the entire time, it also provided insights on what avenues could be pursued to achieve further improvement in the design.

Some possible future research directions for this work are listed below.

5-1 Robustness to model errors and input delays

In this work, we made use of Control Barrier Function (CBF)s to avoid violating our safety requirement. While the CBF-Quadratic Programming (QP) controller design could ensure set invariance if it could operate in continuous time, in practice the controller operates at discrete samples. Since CBFs are a model-based approach, their effectiveness is contingent on the accuracy of the model as well as modelling any delays that exist in the system. While such effects were ignored in this work, it would be prudent to account for them in practice. Recent work in [45] deals with CBFs for sampled-data system with input delays, while work in [29] investigates with the effect of bounded unmodelled disturbances on CBF-QP controllers. Further research in this direction would be beneficial in improving the alignment accuracy of such dual-arm robots.

5-2 Improving the global planner

Most of the effort in this work went into implementing the middle and the lower layer of the hierarchical planner, since the simple nature of the toy example for demonstration allowed for the use of off-the-shelf motion planners from Open Motion Planning Library (OMPL) that have been exposed to MoveIt for the highest layer. In principle, one could use such planners to also solve the problem of imaging the object of interest from multiple poses, by manually querying the planner for point-to-point trajectories. However, it would be desirable to automate this process. Symbolic logic seems to be a promising approach for solving this problem, as shown in the works of [19, 22].

Another point of improvement is in the speed and optimality of the global planning phase. The global planner is currently the bottle neck in the design in terms of time taken from the user sending a request to the planner successfully completing a run. Since the available planning algorithms are probabilistically complete, sufficient planning time needs to be provided to ensure that a solution is found. This problem is exacerbated in terms of probabilistically optimal planners like RRT*, since they tend to use the entire allotted planning time trying to optimize the planned trajectory, unlike non-optimal planners like RRT-Connect which return the first solution they find. This means that considerable time savings can be achieved by improving the implementation of the global planner. One possible avenue to investigate is to have the global planner return the first plan it finds, and iteratively improve the plan during the course of the run. Another more promising avenue is to remove the use of MoveIt's motion planning pipeline and instead directly make use of the planners available in OMPL. This opens up more possibilities, since not all the planners (for example, LTLPlanner and Informed RRT* are currently integrated with MoveIt. This would also allow us to design custom state samplers to speed up the sampling process by exploiting our prior knowledge of the system (for example, making use of a master-slave scheme to increase the probability of sampling valid robot configurations).

5-3 Modifications to the MPC structure

Preliminary results when testing the simulation showed that a two-layer planner (the proposed design without the CBF-QP controller) was capable in keeping the error contained to a surprisingly small bound (within 0.5 mm, well under the design specifications of 3 mm). This would indicate that it would be possible to make further approximations in the internal model and the state constraints in the MPC problem, potentially converting the problem into a QP problem and obtaining savings in computation cost. Some promising avenues of research is in the use of Quasi-Linear Model Predictive Control (qLMPC), as demonstrated in [25, 26], where the nonlinear system and constraints are modelled as Linear Parameter Varying (LPV) systems and constraints using candidate trajectories (i.e., predictions of how the trajectory will evolve in the future). Another possibility is to use the approach in [17], where the model is linearized about a point and feedback from the CBF-QP controller is used to refine the linearization.

A second point of modification would be the collision avoidance constraints. As noted in Section 4-4, employing a Nonlinear Model Predictive Control (NMPC) structure with collision avoidance constraints when the obstacle did not exist in the workspace led to poorer alignment of the End Effector (EEF)s. Therefore, it would be beneficial to change the structure of the NMPC problem based on the number of obstacles in the workspace. One strategy would be to multiplex multiple NMPC instances (each encoding a different number of collision constraints), and using the appropriate instance based on the number of obstacles detected.

5-4 Extension to general collision geometries

In this thesis, we considered obstacles to either be spherical/ellipsoidal, or considered a spherical/ellipsoidal collision box bounding the obstacle. This was in order to make it amenable to the requirement that the CBFs are continuously differentiable. However, this can lead to very conservative CBFs, since the obstacle might occupy only a small fraction of the bounding volume. This makes it difficult to add barriers for self-collisions between the robot arms or with any oblong obstacles. Therefore, it would be desirable to employ more general collision geometries such as swept spheres or polytopes. Research such as the one carried out in [46] gives a promising avenue to implement such features in the future.

Appendix A

Review of the stability of Model Predictive Control

This chapter reviews the conditions under which the stability of suboptimal Model Predictive Control (MPC) can be guaranteed. The discussion in this chapter is adapted from [24].

The standard MPC problem for that is tackled in [24] can be seen in Problem A-1.

$$\begin{aligned} J_n(x_0) = & \\ & \min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} l(x(k), u(k)) + V_f(x(N)) \\ & s.t. \quad x(k+1) = f(x(k), u(k)) \quad \forall k \in \{1, 2, ..., N-1\} \\ & \qquad x(k) \in \mathcal{X} \quad \forall k \in \{0, 1, 2, ..., N-1\} \\ & \qquad u(k) \in \mathcal{U} \quad \forall k \in \{0, 1, 2, ..., N-1\} \\ & \qquad x(N) \in \mathcal{X}_f \\ & \qquad x(0) = x_0 \end{aligned} \tag{A-1}$$

In Problem A-1, $\mathcal{X}$ and $\mathcal{U}$ refer to the set of admissible state and control inputs. We also define the Cartesian product of the two sets $\mathcal{Z} : \mathcal{X} \times \mathcal{U}$. The function $f : \mathcal{Z} \to \mathcal{X}$ represents the discrete time system dynamics. The function $l : \mathcal{Z} \to \mathbb{R}_0^+$ refers to the stage cost and $V_f : \mathcal{X} \to \mathbb{R}_0^+$ is the terminal cost. We define the sequence of states and inputs as $\mathbf{x} = [x(0)\ x(1)\ ...\ x(N)]^T \in \mathcal{X}^{N+1}$ and $\mathbf{u} = [u(0)\ u(1)\ ...\ u(N-1)]^T \in \mathcal{U}^N$ respectively, with $\mathbf{z} \in \mathcal{X}^{N+1} \times \mathcal{U}^N$ being used as shorthand notation for the extended state sequence vector going forward. We use the notation $\phi(k; x, \mathbf{u})$ to denote the state after k time steps when initialized at x and subject to the input sequence $\mathbf{u}$. This allows us to define the optimal cost-to-go function below.

$$V_n(x, \mathbf{u}) = \sum_{k=0}^{N-1} l(\phi(k; x, \mathbf{u})) + V_f(\phi(N; x, \mathbf{u})) \tag{A-2}$$

The first step that is taken is to prove the stability of the system with the MPC controller in closed loop, under the assumption that the optimal solution $\mathbf{z}^*$ to Problem A-1 can be found. There are four key assumptions that are made in [24] that are used to prove the stability of MPC under such a circumstance:

Assumption A.1 (Continuity of system and cost [24]). *The functions $f : \mathcal{Z} \to \mathcal{X}$, $l : \mathcal{Z} \to \mathbb{R}_0^+$ are continuous, $f(0,0) = 0$, $l(0,0) = 0$ and $V_f(0) = 0$.*

Assumption A.2 (Properties of constraint sets [24]). *The set $\mathcal{Z}$ is closed and the set $\mathcal{X}_f \subseteq \mathcal{X}$ is compact. Each set contains the origin. If $\mathcal{U}$ is bounded (hence compact), the set $\mathcal{U}(x)$ is compact for all $x \in \mathcal{X}$. If $\mathcal{U}$ is unbounded, the function $\mathbf{u} \mapsto V_N(x_0, \mathbf{u})$ is coercive, i.e., $V_N(x_0, \mathbf{u}) \to \infty$ as $|\mathbf{u}| \to \infty$ for all $x_0 \in \mathcal{X}$.*

Assumption A.3 (Basic stability assumption [24]). *$V_f(\cdot), \mathcal{X}_f(\cdot)$ and $l(\cdot)$ have the following properties:*

- *For all $x \in \mathcal{X}_f$, there exists a u (such that $(x,u) \in \mathcal{Z}$ satisfying*

$$\begin{aligned} f(x,u) &\in \mathcal{X}_f \\ V_f(f(x,u)) - V_f(x) &\leq -l(x,u) \end{aligned}$$

- *There exist $\mathcal{K}_\infty$ functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ satisfying*

$$\begin{aligned} l(x,u) &\geq \alpha_1(|x|) \quad \forall x \in \mathcal{X}_N, \forall u \text{ s.t. } (x,u) \in \mathcal{Z} \\ V_f(x) &\leq \alpha_f(|x|) \quad \forall x \in \mathcal{X}_f \end{aligned}$$

Assumption A.4 (Weak controllability [24]). *There exists a $\mathcal{K}_\infty$ function $\alpha(\cdot)$ such that*

$$V_N^0(x) \leq \alpha(|x|) \; \forall x \in \mathcal{X}_N$$

It is then proven in [24] that provided these assumptions are satisfied and the global optimum of Problem A-1 can be found, then the system is closed-loop stable.

Proposition A.1 (Asymptotic stability of the origin [24]). *Suppose Assumptions A.1, A.2, A.3 and A.4 are satisfied. Then*

(a) There exists $\mathcal{K}_\infty$ functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ such that for all $x \in \mathcal{X}_N$

$$\begin{aligned} \alpha_1(|x|) \leq V_N^0(x) \leq & \quad \alpha_2(|x|) \\ V_N^0(f(x, \kappa_N(x))) - V_N^0(x) \leq & \quad -\alpha_1(|x|) \end{aligned}$$

(b) The origin is asymptotically stable in $\mathcal{X}_N$ for $x^+ = f(x, \kappa_N(x))$.

The main caveat in such a scenario is that it must indeed be possible to compute the globally optimal solution. This is only possible for Problem A-1 if the dynamics $f : \mathcal{Z} \to \mathcal{X}$ are Linear Time Invariant (LTI) and the sets $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{X}_f$ are convex. If any of these conditions are not, the problem will be rendered non-convex. This means that we will find a local optimum if it exists, but there is no guarantee of finding a global optimum. In such a scenario, it is desirable to know if there are any conditions under which such a suboptimal solution could still guarantee asymptotic stability.

A key step taken in the explanation given in [24] is to consider suboptimal MPC as the evolution of an extended state consisting of the state and the warmstart pair. An admissible warm start must be capable of steering the current state x to the terminal set in a finite horizon subject to input constraints, i.e., $\tilde{\mathbf{u}} \in \mathcal{U}_N(x)$. It must also satisfy the condition that $V_N(x, \tilde{\mathbf{u}}) \leq V_f(x)$, since that ensures that as $|x| \to 0$, $|\tilde{\mathbf{u}}| \to 0$ as well. Therefore, the admissible set of warmstart can be defined as

$$\tilde{\mathcal{U}}_N(x) = \{\tilde{\mathbf{u}} \in \mathcal{U}_N(x) \mid V_N(x, \tilde{\mathbf{u}}) \leq V_f(x) \ if \ x \in \mathcal{X}_f\} \tag{A-3}$$

In case that we are in the terminal set, i.e., $x \in \mathcal{X}_f$, then the terminal control law $\kappa_f(\cdot)$ can also be used to obtain a feasible warm start

$$\tilde{\mathbf{u}}_f(x) := [\kappa_f(x) \ \kappa_f(f(x, \kappa_f(x)) \ ... \ ]^T$$

Given an initial warmstart $\tilde{\mathbf{u}}$, it is then possible to compute a local optimum $\mathbf{u}$, such that $V_n(x, \mathbf{u}) \leq V_N(x, \tilde{\mathbf{u}})$. This allows us to define the set of feasible control sequences $\check{\mathcal{U}}_N(x)$ and the suboptimal control law $\kappa_N(x, \tilde{\mathbf{u}})$ as follows

$$\begin{aligned} \check{\mathcal{U}}_N(x) &= \{\mathbf{u} \mid \mathbf{u} \in \tilde{\mathcal{U}}_n(x), V_n(x, \mathbf{u}) \leq V_N(x, \tilde{\mathbf{u}})\} \\ \kappa_N(x, \tilde{\mathbf{u}}) &= \{u(0) \mid \mathbf{u} \in \check{\mathcal{U}}_N(x)\} \end{aligned}$$

In such a scenario, if the MPC is initialized with a feasible warmstart, then it will return either a more optimal control sequence or the warmstart in the worst case. After the first input is injected, the remaining inputs and the terminal control law can be used to generate the next warmstart for the successor state $x^+ = f(x, u(0))$ as shown below.

$$\tilde{\mathbf{u}}_w(x, \mathbf{u}) = [u(1) \ u(2) \ ... \ u(N-1) \ \kappa_f(x(N))]^T$$

Then, the following function can be used to select an appropriate successor warm start $\tilde{\mathbf{u}}^+$

$$\tilde{\mathbf{u}}^+ = \zeta(x, \mathbf{u}) := \begin{cases} \tilde{\mathbf{u}}_f(x^+) & \text{if } x^+ \in \mathcal{X}_f \text{ and } V_N(x^+, \tilde{\mathbf{u}}_f(x^+)) \leq V_N(x^+, \tilde{\mathbf{u}}_w(x, \mathbf{u}) \\ \tilde{\mathbf{u}}_w(x, \mathbf{u}) & \text{else} \end{cases}$$

With the information written above, [24] proposes an Algorithm to implement suboptimal MPC

Algorithm A.1 (Suboptimal MPC Algorithm [24])**.** *First, choose $\mathcal{X}_f$ and $V_f(\cdot)$ satisfying Assumption 3 and obtain the initial state $x \in \mathcal{X}_n$ and any initial warm start $\tilde{\mathbf{u}} \in \tilde{\mathcal{U}}_N(x)$. Then repeat*

1. *Obtain current measurement of state x.*
2. *Compute any input $\mathbf{u} \in \check{\mathcal{U}}_N(x)$.*

3. *Inject the first element of the input sequence* $\mathbf{u}$.
4. *Compute the next warm start* $\tilde{\mathbf{u}}^+ = \zeta(x, \mathbf{u})$

What is nice about this Algorithm is that it also guarantees that the origin is asymptotically stable.

In order to illustrate this proof, let us define the extended state s as

$$s \in \tilde{\mathcal{S}}_N := \{(x, \tilde{\mathbf{u}}) \mid x \in \mathcal{X}_N, \tilde{\mathbf{u}} \in \tilde{\mathcal{U}}_N(x)\}$$

The evolution of the extended state is governed by the dynamics

$$s^+ \in H(s) := \{(x^+, \tilde{\mathbf{u}}^+) \mid x^+ = f(x, u(0)), \tilde{\mathbf{u}}^+ = \zeta(x, \mathbf{u}), \mathbf{u} \in \check{\mathcal{U}}_N(s)\} \tag{A-4}$$

In order to link the asymptotic behaviour of s with that of x, the following proposition is used in [24].

Proposition A.2 (Linking warm start and state [24])**.** *There exists a function* $\alpha_r(\cdot) \in \mathcal{K}_\infty$ *such that* $|\tilde{\mathbf{u}}| \leq \alpha_r(|x|)$ *for any* $(x, u) \in \tilde{\mathcal{S}}_N$*.*

Proposition A.3 (Global $\mathcal{K}$ function overbound [24])**.** *Let* $\mathcal{X} \subseteq \mathbb{R}^n$ *be closed and suppose that a function* $V : \mathcal{X} \to \mathbb{R}_0^+$ *is continuous at* $x_0 \in \mathcal{X}$ *and locally bounded on* $\mathcal{X}$*. Then, there exists a* $\mathcal{K}$ *function* α *such that*

$$|V(x) - V(x_0)| \leq \alpha(|x - x_0|)\ \forall x \in \mathcal{X}$$

Proposition A.4 (Asymptotic stability of a difference inclusion [24])**.** *If the set* $\mathcal{S}$ *contains the origin, is positive invariant for the difference inclusion* $s^+ \in H(s)$*,* $H(0) = 0$ *and it admits a Lyapunov function* $V(\cdot)$ *in* $\mathcal{S}$*, then the origin is asymptotically stable in* $\mathcal{S}$*.*

We also make use of the following property [24, Equation B.1].

For $\gamma(\cdot) \in \mathcal{K}$*, the following holds for all* $a_i \in \mathbb{R}_0^+$*,* $i \in \{1, 2, ..., n\}, n \in \mathbb{N}$

$$\gamma(na_1) + \ldots + \gamma(na_n) \geq \gamma(a_1 + \ldots + a_n) \tag{A-5}$$

With this, we now have the tools necessary to understand the proof for Proposition 2.1. Take $\tilde{\mathcal{S}}_N$ to contain the origin and to be positive invariant under the dynamics given in A-4 (this happens by construction). If we can show that $V_N(s)$ acts as a Lyapunov function for $s \in \tilde{\mathcal{S}}_N$, then it proves that the algorithm renders the closed loop system asymptotically stable.

By definition,

$$V_N(s) \geq V_N(x, \mathbf{u}) \geq \sum_{i=0}^{N-1} l(x(i), u(i)) \geq \sum_{i=0}^{N-1} \alpha_l(x(i), u(i))$$

By using Equation A-5 and the triangle inequality, we can write

$$\sum_{i=0}^{N-1} \alpha_l(x(i), u(i)) \geq \alpha_l \left(\frac{1}{N} \sum_{i=0}^{N-1} |x(i), u(i)| \right) \geq \alpha_l \left(\frac{1}{N} |(\mathbf{x}, \mathbf{u})| \right) \tag{A-6}$$

By making use of the property that $|\mathbf{x}| \geq |x(i)|\ \forall i \in \{0, 1, \ldots, N\}$, we have that

$$\sum_{i=0}^{N-1} \alpha_l(x(i), u(i)) \geq \alpha_l \left(\frac{1}{N} |(\mathbf{x}, \mathbf{u})| \right) \geq \alpha_l \left(\frac{1}{N} |(x, \mathbf{u})| \right) \coloneqq \alpha_1(|(x, \mathbf{u})|) = \alpha_1(|s|) \tag{A-7}$$

with $\alpha_1 \in \mathcal{K}_\infty$. Furthermore, since we know that the set $\mathcal{S}_N$ is closed and $V_N(s)$ is continuous on $\mathcal{S}_N$, we can conclude from Proposition A.3 that there exists $\alpha_2(\cdot) \in \mathcal{K}_\infty$ such that $V_N(s) \leq \alpha_2(|s|)$ for all $s \in \tilde{\mathcal{S}}_N \subset \mathcal{S}_N$. This establishes that

$$\alpha_1(|s|) \leq V_N(s) \leq \alpha_2(|s|) \quad \forall s \in \tilde{\mathcal{S}}_N,\ \alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty \tag{A-8}$$

This means that $V_N(s)$ is a Lyapunov function candidate.

As in standard MPC analysis, we have for all $s, s^+ \in \tilde{\mathcal{S}}_N$ and $\mathbf{u} \in \check{\mathcal{U}}_N(s)$

$$V_N(s^+) \leq V_N(x, \mathbf{u}) - l(x, u(0)) \leq V_N(x, \mathbf{u}) - \alpha_l(|x, u(0)|)$$

Since $\tilde{\mathbf{u}} \in \check{\mathcal{U}}_N(s)$ by construction, we can write

$$V_N(s^+) \leq V_N(x, \tilde{\mathbf{u}}) - l(x, \tilde{u}(0)) \leq V_N(x, \tilde{\mathbf{u}}) - \alpha_l(|x, \tilde{u}(0)|) \tag{A-9}$$

From the triangle inequality and Proposition A.2, we know that

$$|(x, \tilde{\mathbf{u}})| \leq |x| + |\tilde{\mathbf{u}}| \leq |x| + \alpha_r(|x|) \coloneqq \alpha_{r'}(|x|) \tag{A-10}$$

By making use of the property that $|\mathbf{x}| \geq |x(i)|\ \forall i \in \{0, 1, \ldots, N\}$

$$\alpha_{r'}(|x|) \leq \alpha_{r'}(|x, \tilde{u}(0))) \tag{A-11}$$

This implies that $\alpha_l \circ \alpha_{r'}(|s|) \leq \alpha_l(|(x, \tilde{u}(0)|)$. By defining $\alpha_3(\cdot) \coloneqq \alpha_l \circ \alpha_{r'}(\cdot)$, we can derive from Equation A-9 that

$$V_N(s^+) \leq V_N(s) - \alpha_3(|s|) \quad \forall s \in \tilde{\mathcal{S}}_N,\ \alpha_3 \in \mathcal{K}_\infty \tag{A-12}$$

From Equations A-8 and A-12, it can be concluded that $V_N(s)$ is a Lyapunov function that renders the extended system asymptotically stable. Then from Proposition A.4, it follows that the original closed loop system is also asymptotically stable. Hence it is proved that suboptimal MPC implemented using Algorithm A.1 is able to asymptotically stabilize the system for any $x \in \mathcal{X}_N$.

Appendix B

Test setup details

B-1 Test setup modelling

The setup is modelled with 6 reference frames, with 5 attached to the joints of the robot and the final End Effector (EEF) frame attached to a 6^{th} dummy joint in the center of the ring, as seen in Figure B-1. The axes are coloured as follows: red for x, black for y and blue for z. A circle represents an axis coming out of the plane. The corresponding rotation and translation components of the transformation matrices between subsequent joints is as detailed in Table B-1.

B-2 Servo details

The servo specifications on the Yahboom DOFBOT are as seen in Figure B-2.

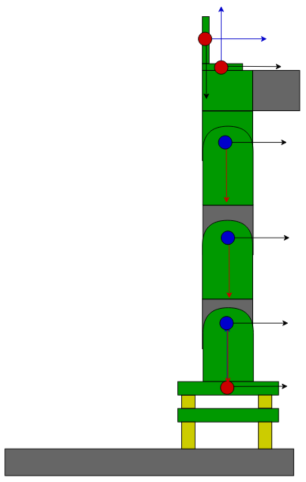
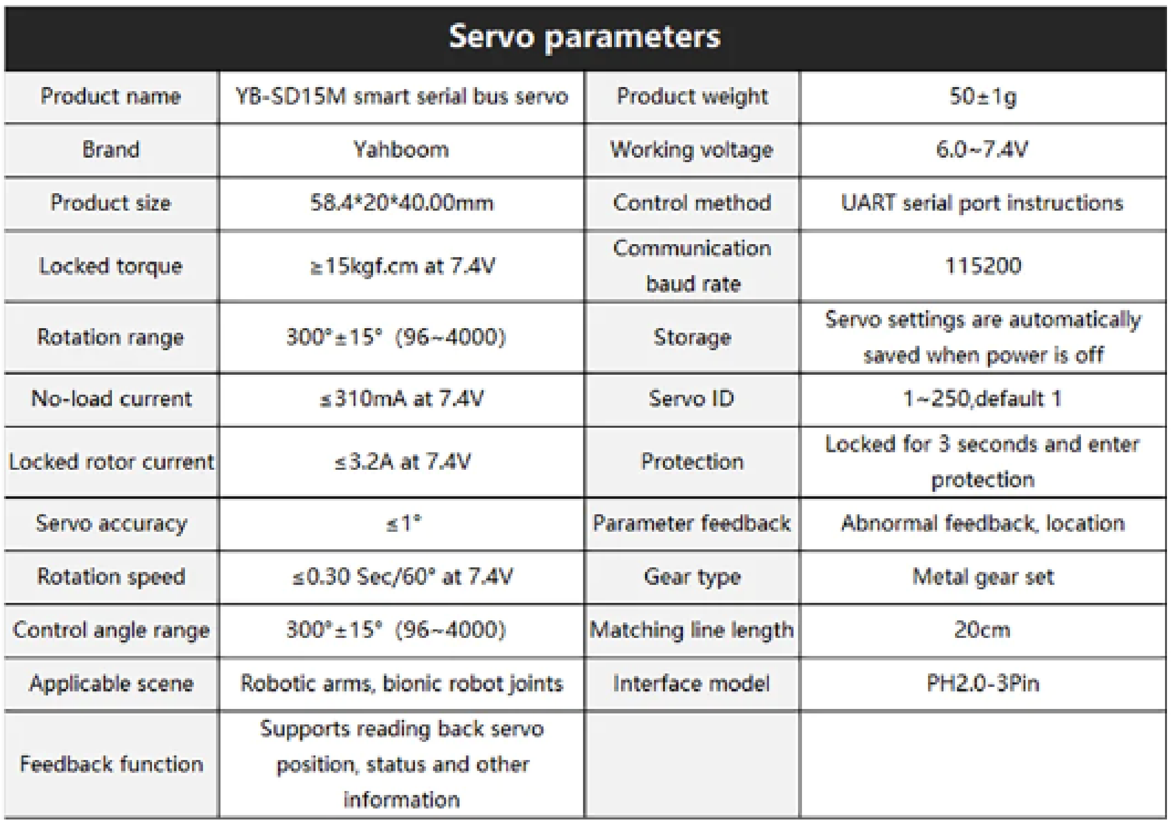


Figure B-1: The different frames of reference used to model the DOFBOT kinematics



Servo parameters			
Product name	YB-SD15M smart serial bus servo	Product weight	50±1g
Brand	Yahboom	Working voltage	6.0~7.4V
Product size	58.4*20*40.00mm	Control method	UART serial port instructions
Locked torque	≥15kgf.cm at 7.4V	Communication baud rate	115200
Rotation range	300°±15° (96~4000)	Storage	Servo settings are automatically saved when power is off
No-load current	≤310mA at 7.4V	Servo ID	1~250,default 1
Locked rotor current	≤3.2A at 7.4V	Protection	Locked for 3 seconds and enter protection
Servo accuracy	≤1°	Parameter feedback	Abnormal feedback, location
Rotation speed	≤0.30 Sec/60° at 7.4V	Gear type	Metal gear set
Control angle range	300°±15° (96~4000)	Matching line length	20cm
Applicable scene	Robotic arms, bionic robot joints	Interface model	PH2.0-3Pin
Feedback function	Supports reading back servo position, status and other information		

Figure B-2: Servo specification sheet [47]

i	**i+1**	${}^{i}R_{i+1}$	${}^{i}t_{i+1}$
0	1	$\begin{bmatrix} c(q_1) & -s(q_1) & 0 \\ s(q_1) & c(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0.064 \end{bmatrix}$
1	2	$\begin{bmatrix} 0 & 0 & 1 \\ s(q_2) & c(q_2) & 0 \\ -c(q_2) & s(q_2) & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0.0435 \end{bmatrix}$
2	3	$\begin{bmatrix} c(q_3) & -s(q_3) & 0 \\ s(q_3) & c(q_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.08285 \\ 0 \\ 0 \end{bmatrix}$
3	4	$\begin{bmatrix} c(q_4) & -s(q_4) & 0 \\ s(q_4) & c(q_4) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.08285 \\ 0 \\ 0 \end{bmatrix}$
4	5	$\begin{bmatrix} 0 & 0 & 1 \\ s(q_5) & c(q_5) & 0 \\ c(q_2) & -s(q_5) & 0 \end{bmatrix}$	$\begin{bmatrix} -0.07385 \\ -0.00215 \\ 0 \end{bmatrix}$
5	6	$\begin{bmatrix} c(q_6) & -s(q_6) & 0 \\ 0 & 0 & 1 \\ -s(q_6) & -c(q_6) & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.011 \\ 0.02 \end{bmatrix}$

Table B-1: Joint transforms for a single DOFBOT arm

Bibliography

[1] A. Ziertmann, P. Jahnke, S. Kerscher, M. Koch, and W. Holub, "Robot guided Computed Tomography: —Production Monitoring in Automotive Industry 4.0," *Journal of the Japan Society for Precision Engineering*, vol. 86, no. 5, pp. 316–322, May 2020. [Online]. Available: https://www.jstage.jst.go.jp/article/jjspe/86/5/86_316/_article

[2] J. A. W. van Pinxteren, "Novel mechatronic architectures for interventional X-ray systems," Phd Thesis 1 (Research TU/e / Graduation TU/e), Technische Universiteit Eindhoven, Eindhoven, Jan. 2021, iSBN: 9789038652023.

[3] A. Fieselmann, J. Steinbrener, A. K. Jerebko, J. M. Voigt, R. Scholz, L. Ritschl, and T. Mertelmeier, "Twin robotic x-ray system for 2d radiographic and 3d cone-beam ct imaging," in *Medical Imaging 2016: Physics of Medical Imaging*, vol. 9783. SPIE, 2016, pp. 128–133.

[4] M. Li, Z. Fang, W. Cong, C. Niu, W. Wu, J. Uher, J. Bennett, J. T. Rubinstein, and G. Wang, "Clinical micro-ct empowered by interior tomography, robotic scanning, and deep learning," *IEEE Access*, vol. 8, pp. 229 018–229 032, 2020.

[5] S. E. Salcudean, H. Moradi, D. G. Black, and N. Navab, "Robot-Assisted Medical Imaging: A Review," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 951–967, Jul. 2022, conference Name: Proceedings of the IEEE.

[6] R. Mohan, "Tip of the Iceberg: Exploring Robot Autonomy for Interventional Healthcare," Phd Thesis 1 (Research TU/e / Graduation TU/e), Technische Universiteit Eindhoven, Eindhoven, Sep. 2020, iSBN: 9789402821567.

[7] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*, ser. Springer Handbooks. Cham: Springer International Publishing, 2016. [Online]. Available: https://link.springer.com/10.1007/978-3-319-32552-1

[8] S. M. LaValle, *Planning Algorithms*, 1st ed. Cambridge University Press, May 2006. [Online]. Available: https://www.cambridge.org/core/product/identifier/9780511546877/type/book

[9] S. LaValle, “Rapidly-exploring random trees : a new tool for path planning,” *The annual research report*, 1998. [Online]. Available: https://www.semanticscholar.org/paper/Rapidly-exploring-random-trees-%3A-a-new-tool-for-LaValle/d967d9550f831a8b3f5cb00f8835a4c866da60ad

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

[11] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “CHOMP: Gradient optimization techniques for efficient motion planning,” in *2009 IEEE International Conference on Robotics and Automation*. Kobe: IEEE, May 2009, pp. 489–494. [Online]. Available: http://ieeexplore.ieee.org/document/5152817/

[12] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574, iSSN: 1050-4729. [Online]. Available: https://ieeexplore.ieee.org/document/5980280

[13] I. Lefkowitz, “Multilevel Approach Applied to Control System Design,” *Journal of Basic Engineering*, vol. 88, no. 2, pp. 392–398, Jun. 1966. [Online]. Available: https://asmedigitalcollection.asme.org/fluidsengineering/article/88/2/392/397240/Multilevel-Approach-Applied-to-Control-System

[14] M. Mesarovic, “Multilevel systems and concepts in process control,” *Proceedings of the IEEE*, vol. 58, no. 1, pp. 111–125, Jan. 1970, conference Name: Proceedings of the IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1449475

[15] W. Findeisen, “Hierarchical Control Systems: An Introduction,” Apr. 1978, num Pages: 105 Place: IIASA, Laxenburg, Austria Publisher: PP-78-001. [Online]. Available: https://iiasa.dev.local/

[16] M. Mesarovic, D. Macko, and Y. Takahara, “Theory of hierarchical, multilevel, systems, ser,” *Mathematics in Science and Engineering, New York: Academic*, vol. 68, 1970.

[17] U. Rosolia, A. Singletary, and A. D. Ames, “Unified Multirate Control: From Low-Level Actuation to High-Level Planning,” *IEEE Transactions on Automatic Control*, vol. 67, no. 12, pp. 6627–6640, Dec. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9802791/

[18] F.-Y. Wang and G. N. Saridis, “A coordination theory for intelligent machines,” *Automatica*, vol. 26, no. 5, pp. 833–844, Sep. 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000510989090001X

[19] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, “Symbolic planning and control of robot motion [grand challenges of robotics],” *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[20] U. Rosolia and A. D. Ames, “Multi-Rate Control Design Leveraging Control Barrier Functions and Model Predictive Control Policies,” *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1007–1012, Jul. 2021, conference Name: IEEE Control Systems Letters. [Online]. Available: https://ieeexplore.ieee.org/document/9137248

[21] A. Dobson and K. E. Bekris, “Sparse roadmap spanners for asymptotically near-optimal motion planning,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 18–47, 2014.

[22] C. I. Vasile and C. Belta, “Sampling-based temporal logic path planning,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4817–4822.

[23] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, “Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 8352–8358, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9561510

[24] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing. [Online]. Available: https://sites.engineering.ucsb.edu/~jbraw/mpc/

[25] G. Cisneros and P. Sebastian, “Quasi-Linear Model Predictive Control: Stability, Modelling and Implementation,” Ph.D. dissertation, Dr. Hut, 2021. [Online]. Available: http://hdl.handle.net/11420/9646

[26] P. S. G. Cisneros, A. Sridharan, and H. Werner, “Constrained Predictive Control of a Robotic Manipulator using quasi-LPV Representations,” *IFAC-PapersOnLine*, vol. 51, no. 26, pp. 118–123, Jan. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318328210

[27] U. Rosolia and F. Borrelli, “Learning how to autonomously race a car: a predictive control approach,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2019.

[28] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, “On the stability of constrained mpc without terminal constraint,” *IEEE transactions on automatic control*, vol. 51, no. 5, pp. 832–836, 2006.

[29] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, “Robustness of control barrier functions for safety critical control,” *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[30] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control Barrier Function Based Quadratic Programs for Safety Critical Systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017, conference Name: IEEE Transactions on Automatic Control. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7782377

[31] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control Barrier Functions: Theory and Applications,” in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 3420–3431. [Online]. Available: https://ieeexplore.ieee.org/document/8796030

[32] Q. Nguyen and K. Sreenath, "Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints," in *2016 American Control Conference (ACC)*, Jul. 2016, pp. 322–328, iSSN: 2378-5861. [Online]. Available: https://ieeexplore.ieee.org/document/7524935

[33] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames, "Safety-critical manipulation for collision-free food preparation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 954–10 961, 2022.

[34] F. Ferraguti, C. T. Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfè, "Safety and Efficiency in Robotics: The Control Barrier Functions Approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, Sep. 2022, conference Name: IEEE Robotics & Automation Magazine. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9788028/references#references

[35] V. Kurtz, P. M. Wensing, and H. Lin, "Control Barrier Functions for Singularity Avoidance in Passivity-Based Manipulator Control," in *2021 60th IEEE Conference on Decision and Control (CDC)*, Dec. 2021, pp. 6125–6130, iSSN: 2576-2370. [Online]. Available: https://ieeexplore.ieee.org/document/9683597

[36] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[37] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, May 2022. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.abm6074

[38] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.

[39] "Hybrid Planning — MoveIt Documentation: Humble documentation." [Online]. Available: https://moveit.picknik.ai/humble/doc/concepts/hybrid_planning/hybrid_planning.html

[40] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[41] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[42] A. Florez, A. Astudillo, W. Decre, J. Swevers, and J. Gillis, "Impact: A toolchain for nonlinear model predictive control specification, prototyping, and deployment," in *Proceedings of the IFAC World Congress 2023*, 2023.

[43] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[44] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: an operator splitting solver for quadratic programs,” *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[45] A. Singletary, Y. Chen, and A. D. Ames, “Control barrier functions for sampled-data systems with input delays,” in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 804–809.

[46] A. Thirugnanam, J. Zeng, and K. Sreenath, “Duality-based Convex Optimization for Real-time Obstacle Avoidance between Polytopes with Control Barrier Functions,” in *2022 American Control Conference (ACC)*. Atlanta, GA, USA: IEEE, Jun. 2022, pp. 2239–2246. [Online]. Available: https://ieeexplore.ieee.org/document/9867246/

[47] “Yahboom 15KG serial bus smart servo and driver debugging board for Robotic Arm.” [Online]. Available: https://category.yahboom.net/products/15kg-serial-bus-servo

Glossary

List of Acronyms

CBF	Control Barrier Function
CLF	Control Lyapunov Function
CT	Computed Tomography
DOF	Degrees of Freedom
ECBF	Exponential Control Barrier Function
EEF	End Effector
FSM	Finite State Machine
LPV	Linear Parameter Varying
LTI	Linear Time Invariant
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
OMPL	Open Motion Planning Library
PRM	Probabilistic Roadmaps
qLMPC	Quasi-Linear Model Predictive Control
QP	Quadratic Programming
ROS	Robot Operating System
RRT	Rapidly exploring Random Trees
SPARS	SPArse Roadmap Spanner
SQP	Sequential Quadratic Programming
URDF	Unified Robot Description Format