GPU-accelerated Real-time Motion Planning for Safe Human-Robot Collaboration

Shohei Fujii

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

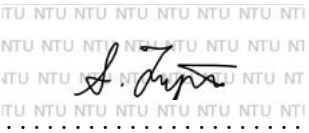
2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

21 July 2023

Date



Shohei Fujii

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

21 July 2023

.....

Date

.....

Prof. Pham Quang Cuong

Authorship Attribution Statement

This thesis contains material from 2 papers accepted at conferences in which I am listed as the first, corresponding author.

Chapter 3 is published as Shohei Fujii, and Quang-Cuong Pham. Realtime Trajectory Smoothing with Neural Nets. International Conference on Robotics and Automation (ICRA), 7248-7254 2022. DOI: 10.1109/ICRA46639.2022.9812418.

The contributions of the co-authors are as follows:

- Prof Cuong provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Prof. Cuong.
- I designed the study and performed all experiments.

Chapter 4 is published as Shohei Fujii, and Quang-Cuong Pham. Time-Optimal Path Tracking with ISO Safety Guarantees. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2023, DOI: 10.1109/IROS55552.2023.10342287.

The contributions of the co-authors are as follows:

- Prof Cuong provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Prof. Cuong.
- I designed the study and performed all experiments.

Chapter 5 is accepted for publication as Shohei Fujii, and Quang-Cuong Pham. Real-time Batched Distance Computation for Time-Optimal Safe Path Tracking, International Conference on Robotics and Automation (ICRA), 2024.

The contributions of the co-authors are as follows:

- Prof Cuong provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Prof. Cuong.
- I designed the study and performed all experiments.

21 July 2023

. . . .

Date

. . . .

Shohei Fujii

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Pham Quang-Cuong. I am deeply grateful for his patience, guidance and encouragement throughout my Ph.D. study. His invitation to join his lab was a turning point in my life, as it truly changed my life for the better. Without his invaluable support, I would not have been able to embark on my Ph.D. journey. He has consistently been there for me, providing support not only in research but also in my personal life. His energetic and positive attitude towards research has been a constant source of inspiration and will continue to guide me throughout my life.

At my Qualification Examination, the advice in the meeting with Dr. Lum Guo Zhan, Assistant Professor at NTU and one of my TACs, had a profound impact on me. His insightful comment on the safety of my work motivated me to deeply consider the safety of robots in a collaborative environment and spurred me to work on ensuring the safety of robotic motion.

I am also grateful to all of my lab members, including Dr. Bing Song, Dr. Kai Pfeiffer, Dr. Van-Thach Do, Joyce Lim Xin Yan, Vuong Quoc Nghia, Nicholas Adrian, Shan Shilin, Chen Qi Jing, Leonardo Edgar and many others. They provided immense support to me when I started my Ph.D. program remotely from Japan during the covid-19 crisis, and they warmly welcomed me to the lab upon my arrival in Singapore. Their daily support and encouragement have been invaluable to me. The daily discussion with them in the lab has always been a great source of inspiration and motivation for me. I want to express my sincere gratitude to Kai Pfeiffer and Bing Song for their diligent proofreading of my thesis. Their contributions have improved the quality of my work. Thanks to all of them, my time in Singapore during my Ph.D. study has become truly memorable.

Additionally, I would like to express my gratitude for the outstanding support and assistance from DENSO Japan and DENSO International Asia. They have provided me with this great opportunity to focus on my work and study in Singapore, as well as supported the publication of my research results.

Finally, I would like to extend my appreciation to my parents, grandparents and my brother for their unconditional love and support in my entire life. Last but certainly not least, I would like to thank my wife for accompanying me all the way to Singapore and offering her unwavering support at all times.

Abstract

Robotic automation has a significant role in industry over decades. As the demand for complex tasks increases, there has been a recent anticipation for robotic automation in human-robot collaborative environments, leading to the introduction of commercial collaborative robots. However, current robot controllers reduce the speed of robots to secure the safety of humans, which results in conservative behavior and lower performance with collaborative robots. A challenge in this thesis is to *maximize the productivity of collaborative robots while ensuring safety*, aiming for productivity of robots under collaborative situations comparable to that of traditional robots.

In our first work, we introduce a rapid trajectory smoother, primarily to enhance productivity. Existing real-time path planners lack the smoothing post-processing step – which is crucial in sampling-based motion planning – resulting in the trajectories being jerky, and therefore inefficient and less human-friendly. Our rapid trajectory smoother, based on a shortcutting technique, leverages fast clearance inference by a novel neural network and can consistently smooth a trajectory for a 6 DoF robot within 200 ms on a commercial GPU. A comparison shows that our smoother is faster than the state-of-the-art method and the smoothed trajectory is more efficient than the original jerky trajectory even when considering the time required for smoothing.

Subsequently, we propose a time-optimal safe path tracking algorithm, with a particular focus on ensuring safety. Our path tracking algorithm is formulated based on Time-Optimal Path Problem based on Reachability Analysis (TOPP-RA) and proven to provide the fastest control policy for controlling a robot to track a given path. Our method guarantees the safety of human operators in the sense that *the robot will collide only when the robot has a zero velocity*, in accordance with ISO safety standards. We also demonstrate the application of our method in a 6-DoF industrial robot scenario.

Another challenge is that, to achieve *true* time-optimality in safe path tracking, it is crucial to have precise distances between obstacles and a robot at waypoints along an executing path. However, existing methods for computing distances between a robot and obstacles are either too slow for real-time applications, or inaccurate for achieving time-optimality. Thus, we propose a batched distance checker for time-optimal safe path tracking. Our method can evaluate distances of a trajectory in less than 1 millisecond on GPU at runtime, making it suitable for time-critical robotic control. We experimentally demonstrate that our method can navigate a 6-DoF robot earlier than a geometric-primitives-based distance checker in a dynamic, collaborative environment.

Throughout this thesis, we emphasize the performance of our algorithms and their implementations. Since our focus is on industrial applications, algorithm performance is critical for the practicality of our methods. Parallelization plays an important role in achieving high performance, especially with the widespread and powerful GPUs. Therefore, in addition to explaining the proposed algorithms, we develop and benchmark our GPU-accelerated implementations.

We hope that this thesis will pave the way for further development and application of human-robot collaboration both in industry and beyond.

Contents

Chapter 1

Introduction

1.1 Background

Since the emergence of industrial robots, robotic automation has played a pivotal role in driving the growth of various industries such as automotive, electrical/electronics, semiconductors, and manufacturing. The industrial sectors and robotics have mutually evolved and progressed through their interconnected relationship. Those industries, which involve relatively simple and mechanized tasks, have witnessed extensive adoption of robotic automation. The market size of robots continues to expand, and despite the impact of the COVID-19 pandemic and geopolitical tensions on new installations in 2020, the demand for robots has shown a resurgence in 2021 [1].

In these traditional robotic applications, robots simply repeat their work within a fixed environment. System integrators initially teach the robot's motions, which are then executed to perform simple and repetitive tasks. In the case of industrial robot manipulators, for example, integrators teach waypoints that a robot passes and stops at beforehand. The robot follows these pre-defined waypoints to complete its tasks such as pick-and-place operations from points A to B, precise parts assembly, welding and spray painting along pre-defined paths. Similarly, for the automation of transportation and delivery within a plant using Automated Guided Vehicle (AGV), system integrators place tapes on the ground to connect specific locations and set markers at start and goal points. AGVs then track those tapes to carry objects autonomously. This *teach-and-playback* method is widely used at the early

stages of Mass Production and Automation in Industry 3.0 [2]. In this scheme, the speed and productivity of the robotic system are the primary concern. As the speed increases, the productivity rises, resulting in higher profits. The robot's motion can be thoroughly optimized offline to the maximum of the capability of robots.

However, things have changed in modern applications. Tasks have become complex and dynamic, and the robots' environment is no longer fixed. Simple and repetitive motions are not sufficient to accomplish these tasks. Accordingly, robots have started to operate side-by-side with human workers with human support. This collaborative approach introduces another dynamism in the scene, posing challenges that robots have not encountered before. For instance, in assembly lines, human operators need to share the workspace with robots to handle the parts that robots are not able to, such as flexible cables. Similarly, the demand for faster transportation and more flexible delivery requires AGVs to dynamically compute and adjust their routes in real-time, avoiding other robots, obstacles and even humans. Safety of surrounding human workers becomes a crucial concern, leading to online, heavy computational requirements. As exemplified above, on top of the speed and productivity as demanded before, modern robots are required to dynamically compute their behavior to fulfill their tasks while guaranteeing the safety of surrounding humans.

Obviously, the safety of humans surrounding robots is the primary concern in such circumstances [3]. As the demand for human-robot interactive situation grows, the International Organization for Standardization (ISO) standardizes technical requirements for safety. ISO 3691-4 [4] claims that AGV must control its speed and brake in case of emergency, categorizing operating zones and confined zones. ISO/TS 15066 [3, 5] regulates speed/velocity and force/torque of collaborative robots to prevent hazardous motions. Specifically, ISO/TS 15066 defines a Speed and Separation Monitoring (SSM) framework and a Power and Force Limiting (PFL) framework as a third and fourth category respectively. In SSM, a robot's speed must stop when the distance between the robot and human workers becomes a specified protective distance. On the other hand, in PFL, a robot must be designed to be safe inherently or systematically by reducing the force it can apply in the physical contact below a threshold limit. These standards describe methods to prevent or reduce risks in a collaborative situation, providing a guideline for robot

manufacturers and system integrators to design and deploy safe robots. They are recommended, or required in some countries such as Japan, to follow these specifications when implementing robots in collaborative situations.

Indeed, many commercial collaborative robots, or *cobots*, that are certified with ISO standards and/or their equivalent local safety standards, have been released. A range of applications of these robots have been reported, including material handling, welding, assembly, and automotive [3]. However, the adoption of collaborative robots remains limited [1]. Collaborative robots do not prevail in industry as they were expected despite their potential capability and flexibility, due to safety concerns [6]. The authors of [7] discuss the reasons of the unpopularity of cobots by conducting interviews with stakeholders in industry. They argue that safety concerns diminish both productivity and flexibility of cobots; each application of cobots brings new safety problems and necessitates a risk assessment in accordance with the safety standards. Hence, many stakeholders use cobots similarly to traditional robots, without safety fences, and compare their performance with that of traditional robots with larger capacities. This comparison creates the impression that cobots are not cost-effective. As a result, cobots are concluded as "slow and expensive" by them. Collaborative robots are trapped in a pitfall between productivity and flexibility due to safety problems, which prevents the collaborative robots from being widely deployed. We would like to quote the following honest phrase by one of the interviewees from [7]: 'In theory, the robot could find a screw head using these new (collaborative) technologies, the problem is that it is very time consuming [...] and time is money.'

From a system engineering perspective, several methodological approaches have been introduced to mitigate this issue [7, 8]. Their approach helps categorize collaborative situations that robot users encounter and analyze their required specifications. However, they merely suggest one existing technology to *manage* the situations, rather than directly *solving* the problem itself. We, instead, technically tackle against this fundamental and critical issue. Our main challenge is to *maximize the productivity of cobots with safety guarantee*. We believe that collaborative robots have the potential to match the capabilities of traditional robots even under safety regulations and that this is the key to bringing true robotic intelligence into real applications both in and outside the industry.

To address the challenge above, breakthroughs are needed in multiple fronts of robotics such as control, *motion planning*, manipulation, task planning, vision, and human-robot interface. Among these, we focus on enhancing the productivity of collaborative robot manipulators in the field of *motion planning* under safety constraints. Specifically, we concentrate on how to navigate a robot faster even in a dynamic environment where humans and robots coexist. In the following, we discuss the problems of robotic motion planning in the context of seeking both productivity and safety in collaborative scenarios.

1. Trajectory planning problem – How to compute a smooth, collision-free trajectory rapidly and repeatedly in a dynamic environment?

   First of all, a motion planner is required to explore the configuration space of a robot system and plan a collision-free motion. The computed motion needs to be feasible, smooth and short to enable the robot to follow efficiently. Moreover, since the environment is not static, the planner needs to re-plan a trajectory at runtime to avoid collision with moving obstacles. The re-planned trajectory needs to be consistent with the executing motion. The key challenge is to plan a *short and smooth* collision-free trajectory *rapidly* at runtime. Jerky motion leads to inefficiency and less human-friendliness [9]. The computational speed of the planner is also crucial to facilitate online re-planning. Thus, both smoothness and speed are essential in the trajectory planning.

   Another challenge is that industrial situations lack prior knowledge about obstacles such as their shape, size and position. Human operators occasionally hold various tools and parts in their hands, that cannot be generally tracked without additional costs. Therefore, the planner needs to be able to handle un-modeled obstacles in the environment.

2. Velocity planning problem – How to obtain an optimal and safe velocity profile along the trajectory in real-time in a dynamic environment?

   Secondly, a controller finds the fastest way to traverse the trajectory while respecting the system constraints. Although the planned trajectory is collision-free at the time of calculation, the trajectory can be interfered with by a human during its execution. Due to the large dimensionality of the problem, the trajectory planner has difficulty in ensuring human's safety, reflecting the

real situation in real-time. Besides, in many applications, such as welding and painting, the path is fixed. In such scenarios, we can only control the velocity along the path.

Here, we consider velocity planning problem. The robot tracks the path with a certain velocity profile, ensuring both absolute safety *and* time-optimality to maintain its productivity. The velocity planning problem is also known as *Time-Optimal Path Parameterization* (TOPP) problem, that was first introduced three decades ago [10], and a number of methods have been proposed for TOPP. However, the existing methods for TOPP assume a static environment and do not consider any safety constraints.

In general, it is impossible to avoid all human-robot collisions. For instance, even when the robot does not move at all, a human operator can still collide with it by hitting it of her own voluntary motion. However, in the SSM framework of ISO standards, it is possible to minimize harm by requiring this: *if* a collision ever occurs, then the robot must be in a *stationary state* (all links have zero velocity) at the time instant of the collision [5]. This is intuitive because the kinetic energy is proportional to the square of velocity, and human's injury must be avoided at a collision with a robot that has zero kinetic energy.

The key challenge is to compute the *optimal* and *safe* solution in real-time under a dynamic environment. Here, *safe* is in the meaning that 'the robot must stop when the minimum distance between the robot and obstacles is zero, or under a given protective distance'. Once again, prior knowledge about obstacles is not available in this problem either.

1.2 Contributions

We address the aforementioned problems in this thesis. The first work primarily focuses on enhancing productivity concerning the trajectory planning problem. Our second work takes a closer look at safety and aims to find the optimal velocity profile for a given path under safety constraints. The third part of this thesis seeks a truly optimal solution in velocity planning through precise distance computation, in combination with the second work.

1.2.1 Rapid Trajectory Smoothing with Neural Nets

Real-time Motion Planning (RMP) has been proposed to enable human-robot collaboration [11, 12]. In RMP, obstacles are detected in real time through a vision system, and new trajectories are planned with respect to the current positions of the obstacles, and immediately executed on the robot. Existing real-time motion planners, however, lack the smoothing post-processing step – which is crucial in sampling-based motion planning – resulting in the planned trajectories being jerky, and therefore inefficient and less human-friendly [13, 14]. The first contribution of this thesis is a Rapid Trajectory Smoother based on the shortcutting technique to address this issue. Leveraging fast clearance inference by a novel neural network, the proposed method is able to consistently smooth the trajectories of a 6-DoF industrial robot arm within 200 ms on a commercial GPU. We integrate the proposed smoother into a full Vision–Motion Planning–Execution loop and demonstrate real-time, smooth, performance of an industrial robot subject to dynamic obstacles.

1.2.2 Real-time Time-Optimal Path Tracking with ISO Safety Guarantees

The second contribution of this thesis is a time-optimal control policy based on Time-Optimal Path Parameterization via Reachability Analysis (TOPP-RA) [15], guaranteeing safe behavior in the velocity planning problem. Specifically, we prove that: for any robot motion that is strictly faster than the motion recommended by our policy, there exists a human motion that results in a collision with the robot in a non-stationary state. Correlatively, we show that, in simulation, our policy is strictly less conservative than state-of-the-art safe robot control methods. In addition, we propose a parallelization method to reduce the computation time of our pre-computation phase (down to about 0.5 sec, practically), which enables the whole pipeline (including the pre-computation) to be executed at runtime, nearly in real-time. Finally, we demonstrate the application of our method in a scenario: time-optimal safe control of a 6-DoF industrial robot.

1.2.3 Real-time Batched Distance Computation based on Link-local Signed Distance Fields

Time-optimal safe path tracking algorithm mentioned above has a missing piece to achieve true optimality in path tracking. To evaluate the *safeness* of waypoints along the entire executing path, it is necessary to compute distances between obstacles and a robot at each waypoint. Currently, a naive distance checker based on geometric primitives (spheres) is used for fast computations. However, this method returns conservative and shorter distances than their actual values, resulting in a conservative behavior of the robot. On the other hand, an exact mesh-to-mesh geometric distance checker provides precise results but is too slow to evaluate in real-time [16]. Therefore, a new approach is needed to compute distances: 1. at many robot configurations, 2. in real-time for online robot control, 3. as precisely as possible for optimal control. The third contribution is a batched, fast and precise distance computation method based on precomputed link-local Signed Distance Fields(SDFs). Our method can check distances for waypoints along an executing trajectory within less than 1 millisecond on GPU at runtime, which is suitable for time-critical robotic control. Additionally, a neural approximation has been proposed to accelerate the preprocessing process by 2x. Finally, we experimentally demonstrate that our method can navigate a 6-DoF robot earlier than a geometric-primitives based distance checker in a dynamic, collaborative environment.

Throughout this thesis, we place a strong emphasis on the performance of the proposed algorithms and their implementations. Indeed, as the topic of this thesis – industrial robotics – is closely related to applications, algorithm performance is critical for its practical significance. The key to algorithm performance is the use of parallelization, which can dramatically improve the running speed, especially with the availability of powerful GPUs. Therefore, in addition to describing and theoretically analyzing the proposed algorithms, we develop and benchmark our GPU-accelerated implementations.

1.3 Outline of the Thesis

Related work are reviewed in Chapter 2. Chapter 3, Chapter 4 and Chapter 5 shows our contributions consecutively. Conclusion and future perspective are discussed in Chapter 6.

Chapter 2

Literature Review

In this chapter, we review the literature related to motion control and perception of autonomous collaborative robots. We also discuss the unique contributions made by this thesis in relation to existing research (Fig. 2.1).

2.1 Pre-Collision Safety and Post-Collision Safety

Safety of humans is the essential concern with autonomous collaborative robotics. Safety implementations are classified into two strategies: Pre-Collision Safety Strategy and Post-Collision Safety Strategy [17–19]. Pre-Collision Safety Strategy is a technique to ensure the safety of robots before collisions, for example, by modifying robot's trajectory to avoid collision or decreasing robot's speed, while Post-Collision Strategy is to mitigate human's injury induced by the collision by reducing the forces the robot can apply using tactile sensors, inertial sensors and compliance controls.

Most of the commercial collaborative robots available implement Post-Collision Safety Strategy [3]. After the release of the world's first commercial collaborative robot, LBR 3, from KUKA in 2004 [20], various collaborative robots have been released from different companies such as UR-series from Universal Robots, CR-series from FANUC, YuMi from ABB, Franka Arm from Franka Emika, and LBRA iiwa series from KUKA [3]. This is presumably because they do not require external sensors such as cameras and can be implemented only with the embedded

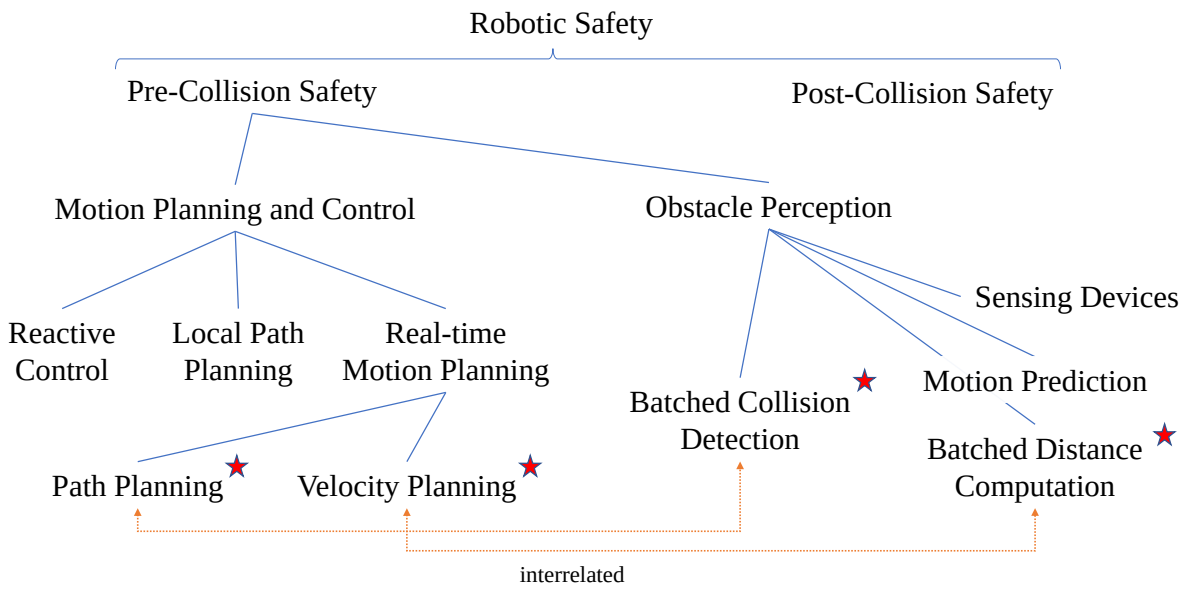


FIGURE 2.1: Hierarchical structure of the sections of literature review. Our contributions are presented within the components marked with stars.

torque/current sensors or tactile sensors, allowing manufacturers to sell them as they are without additional sensory accessories. he safety regulation in ISO standards that corresponds to Post-Collision Safety Strategy is Power and Force Limiting (PFL) [5], which regulates a robot's force and torque in contact with humans. The robot must be designed to be safe inherently or systematically by controlling the force the robot applies at the physical contact below a threshold limit. The commercial robots should (or must, in some countries such as Japan [21]) follow and be certified with ISO or its equivalent standards.

Post-collision safety is fundamental for human-robot interaction, including various research areas such as collision detection and reaction [19], danger mitigation by an evasive motion [22], contact force estimation and interactive control [23, 24]. However, we suppose there is limited potential to improve the performance of robots only through this strategy, due to the following characteristics [25]: Firstly, robots must move slowly and carefully enough to safely collide with humans, which reduces their overall performance. Secondly, a collision itself is not inherently safe, especially when robots hold an sharp and edgy object. The risk involved with such collisions necessitates precautionary measures to avoid potential harm and makes

risk assessments difficult in industrial settings. Additionally, collisions may lead to decreased payload of robots, limiting their potential applications. Thirdly, collisions disturb the human's task and are both physically and psychologically harmful for human workers, making the robots economically disadvantageous [26]. Instead, in this thesis, we focus on measures for the robot to respond before such a collision occurs. We assume that in collaborative industrial situations, robots can move freely as long as collisions are avoided. In the event of a collision, there should be no impact on the human. We believe that further improvements in the performance can be better achieved by addressing pre-collision situations. Therefore, our contribution aims for Pre-Collision Safety Strategy.

It is worth mentioning that the Pre-Collision Safety Strategy does not require dedicated robots. We can implement pre-collision safety strategy for traditional industrial robots by combining them with external sensors, which is cost-effective and economical especially for the users who already own their robots.

In the following sections, we focus on Pre-Collision Safety Strategy. Firstly, we describe Motion Planning and Control in Section 2.2. Specifically related to our work, we review the techniques to generate robot's trajectory to avoid collision (Motion Planning) in Section 2.2.4.1 and correctly control the speed of robot (Velocity Planning) in Section 2.2.4.2. Secondly, we review the techniques for obstacle perception which is indispensable to foresee possible collision.

2.2 Motion Planning and Control

2.2.1 Reactive Control

Reactive control is a control strategy that controls a robot at every control cycle and the robot reacts to changes in the environment. One of the common approaches is 'Artificial potential field'. Artificial potential field is usually a function composed of two potentials: one is an attractive field towards a goal and the other is a repulsive field away from obstacles. The gradient of the weighted sum of the two potentials navigates a robot towards its goal avoiding collision with obstacles. It is computationally lighter compared to graph-search algorithms and has real-time capability.

Traditionally, a fundamental contribution is introduced in [27, 28]. However, although the basic approach can generate a smooth and evasive motion, the potential field has a local-minima problem, where a robot can get stuck and trapped in it before reaching its target [29]. Therefore, many methods have been proposed that can solve the local-minima problem [30–32]. Harmonic potential field (HPF) approach is one of such methods [33, 34]. HPF is designed not to have a local minimum by constraining the differentials of the potential field to satisfy a Laplace equation. This approach is proven to guarantee the robot to reach its goal and is suitable for motion planning. However, it requires significant computation to 'propagate' the information of obstacles in the environment to all over the workspace, which is not satisfactory for planning and control under a dynamic environment. HPF is still an active research topic [35, 36] Other than HPF, there are many variants recently introduced such as Electromagnetic Fields [37], Gyroscopic Force [38] and Circular Fields [39–41]. However, the computation of potential field is still a bottleneck of this type of approaches, otherwise they restrict its exploration scope and plan a path locally rather than globally [39]. The idea of safety is considered in reactive control. One of the examples is 'danger index', where a danger field is proposed to measure how dangerous an robot is based on its current position and velocity with respect to objects [42, 43]. 'Kinetostatic safety field' [44] is another approach that measures the safety of obstacles based on the distance between each of them and their linear link velocities. By analytically constructing a closed-form polynomial function of the kinetostatic safety field beforehand, the computation of the safety field is dramatically accelerated and takes $1 \sim 2$ µs. However, this work requires mesh models of the robot and the obstacles including humans in advance and those need to be tracked, which is not always feasible in the real world. With the advance of deep learning, there exist an interesting approach that uses deep neural network for learning the mapping from pointcloud to 'cost-to-go' function [45]. The 'cost-to-go' function is directly constructed from the pointcloud by the pre-trained neural network, and its evaluation speed is fast by leveraging GPU. The function navigates a robot towards a goal in its configuration space as with the artificial potential field. Though it has a problem in generalizability and safety, it would be a promising approach in the future.

Another research stream of reactive control is Inverse Kinematics (IK) with collision avoidance. It is especially useful when a robot needs to follow a trajectory of its end-effector in a Cartesian space. Safe physical human-robot interaction is achieved

in [46] which combines collision detection, robot reaction and collision avoidance in a safe manner using Kinect depth sensor. Collision IK [47], the successor of Relaxed IK [48], is a technique to compute inverse kinetics while considering the continuity of the solution to the current robot joint values and avoiding obstacles. Another research works on reactive inverse kinematics, where a robot follows its target while suppressing overshoot of the motion by applying safety margin dynamically without any optimization [49]. This approach can control the robot quickly and adaptively without any collision with obstacles.

There exist another approach which formulates reactive collision avoidance as a convex optimization problem using simplified convex geometric models for a robot and obstacles [50]. Such an optimization-based approach has evolved to a more generic (local) motion planner; see Section 2.2.3.2 and Section 2.2.4.1.

We do not select the reactive strategy for two main reasons as pointed out in [25]: Excessive reaction caused by repulsive velocities can reduce the productivity of the robot; and the rapid and sudden evasive motion scares human workers and makes the robot's behavior less human-friendly.

2.2.2 Learning-based Policy

The recent success of applying reinforcement learning (RL) to robotics such as drone racing [51] and quadruped robots [52, 53] has increased our expectations for its potential in other fields. However, the main focus of the RL community for robotic manipulator is on object manipulation and a relatively limited amount of research has been conducted on RL for collision avoidance in a human-robot setting. The application of RL combined with a path planner (PRM [54]) can be found early in 2003 [55]. The authors use a basic Q-learning [56] to find a discrete collision-free configuration to avoid dynamic obstacles projected in a configuration space. The search space is limited to 3D to make computation and training easier. Recent approaches combine RL and a motion planner and use the learned policy as an efficient explorer in a high-dimensional configuration space while locally avoiding collision with obstacles in a sophisticated way [57, 58]. The usage of modern RL for reactive control of a robotic manipulator first can be found in [59]. The learned policy directly controls a 6 DoF robot manipulator and successfully enables it to move away from a single randomly-moving sphere while tracking a target point.

Later, the authors of [60] succeed to train a policy for a robotic manipulator to avoid a moving voxelized human arm. There exists an attempt to use RL for learning and finding an avoidance action in the null space of Jacobian matrix of a redundant manipulator [61]. In terms of safety issue, a learned behavior verifier and a hazard estimator are proposed respectively in [62, 63]. Since, the previous approaches do not guarantee the safety, a *safety shield* concept is recently proposed that provably guarantees human safety even with RL [64].

We do not deal with a RL-based method in this thesis because RL has disadvantages for industrial applications and is still at an early stage of research; for example, RL still lacks generalizability; training of a model does not always successfully run for any kind of robots and tasks, and requires a skilled engineer to tune the hyperparameters; transfer learning does not necessarily work either; and so on. However, we also believe RL will be one of the promising approaches for collaborative robots in the future.

2.2.3 Local Path Planning

Local Path Planning is a problem of finding a collision-free path given a reference trajectory. The reference trajectory is usually given by a higher-level planner such as a task planner or a motion planner described later in Section 2.2.4.1.

2.2.3.1 Model Predictive Control

Model Predictive Control (MPC) formulates a local path planning problem as an optimization problem with kinematic/dynamic motion prediction based on a given model of the system. For every control loop, the algorithm predicts the future trajectory of a robot for a finite time span, finds the best control inputs for the time span and applies the first element of the optimal control input to the system repeatedly [65]. MPC is also called as "Receding horizon control".

Collision avoidance is the source of difficulty in the optimization of MPC. High-frequent controller for nonlinear model predictive control of a robotic manipulator has already been proposed [66] based on Differential Dynamic Programming (DDP) [67], however the collision avoidance term introduces another nonlinearity and non-convexity into the problem, which makes it hard to solve in real-time and

trapped in local-minima [68]. Approximate MPC io another attempt to speed up the MPC computation, where a trained neural network directly and implicitly give a solution to the original robust MPC problem [69]. Still, the trained network cannot handle the collision avoidance problem with dynamic obstacles.

Majority of MPC is solved as a numerical optimization. Instead of time-consuming numerical differentiation in the optimization, symbolic differentiation is adopted with the Casadi framework [70]. The extra computation for differentiation is negligible for static environments, but it is not for dynamic problems because differentiation needs to run at runtime [71]. HyperGraph formulation can dramatically mitigate the extra preparation time by exploiting the sparse structure in MPC [71], but it makes the runtime performance worse compared with that of no sparsity exploitation, and importantly, it cannot still handle un-modeled obstacles [72]. Sampling-based MPC such as Model-Predictive Path Integral (MPPI) [73] and Cross Entropy Method (CEM) [74] are another promising solution to this problem. Highly performative MPC has been proposed in [75]. The authors parallelize action sampling and forward dynamics prediction on Graphics Processing Unit(GPU) and accelerates collision detection with a neural network. However, exploration in the action space is still computationally expensive due to its high dimensionality and does not offer a good collision avoidance behavior. Although its real-time capability reaches 125 Hz, their collision avoidance capability is still limited even against a single obstacle due to its difficulty in sampling.

2.2.3.2 Instantaneous Robot Control

Instead of handling the computationally heavy non-linear MPC directly, researchers have linearized and re-formulated the problem into Quadratic Programming (QP) for real-time capability [76]. It is designed with a quadratic cost function and linear inequality constraints. Tsai et al. [77] take a receding horizon based approach like MPC and formulate a QP problem to find a collision-free path online in a dynamic environment. Robots and obstacles are simplified with ellipsoids, jacobians are approximated by the first order Taylor expansion and they integrate several safety indices for collision avoidance in a quadratic format. They eventually observe more than 1000x speedup compared to a NLP-based approach in an experiment with a simulated 2D robotic manipulator. Ragaglia et al. [25] formulate a QP problem for a path tracking problem where a robot tries to follow its pre-defined

path to the best of its ability while satisfying safety constraints. The humans and robots are modeled with line segments (beams), and the robot successfully alters its programmed trajectory reactively and safely in a collaborative environment, and its computation time is within 4 ms. However, these methods require geometric and simple models of robots and obstacles, which necessitates a human skeleton tracking that may not be always available in real-world applications. Due to their formulation, the number of inequality constraints increases exponentially as the number of obstacle increases. Thus, they cannot directly handle sensory input such as pointcloud or voxels.

Recent attempts for collision avoidance comes from control theory; the control invariant set and *Control Barrier Functions* (CBF) [78]. The safety control problem is formulated as a QP which has inequality constraints for safety ensuring that the distance from obstacles is greater than zero [79]. The inequality constraints are asymptotically and quickly satisfied at runtime and can guarantee collision-free control input. QP itself has been well-studied and many fast solvers have been developed commercially and in open-source, therefore CBF-based controller may also be one of the promising solution for real-time robotic control with safety guarantee [80]. However, to convert dynamic obstacles into inequality constraints, points are sampled from the surface of a robot and an environment to retrieve distances between them, which makes the problem unstrict [81]. Moreover, to realize its real-time capability, CBF is relaxed with a limited number of point sampling, resulting in 'probabilistic' safety guarantee [81]. Further work will be necessary to handle dynamic obstacles while 100% guaranteeing safety.

In summary, local path planning handles both motion planning and time parameterization (see Section 2.2.4.2) as an optimization problem simultaneously without decomposing them though it is done for a short horizon. Therefore, it is inherently difficult and computationally expensive.

2.2.4 Real-time Motion Planning

Motion planning is a problem of finding a collision-free path given a starting point and a goal point. This has a long history of research since 1980s [82]. Shiller et al. [83] propose a fundamental approach that decomposes a time-optimal collision-free motion finding problem into a collision-free path planning problem (PPP) in a

configuration space and a time-optimal velocity planning problem (VPP) along the path. As sampling-based planners such as Rapidly-exploring Random Tree (RRT) [84] and Probabilistic Roadmaps (PRM) [54] are proposed to solve the collision-free path planning problem, post processors of these planners are required and proposed to shorten the generated trajectory and makes it smooth and feasible for a robot. Consequently, optimization-based planners such as CHOMP [85] and TrajOpt [86] are proposed which directly smoothen the trajectory via numerical optimization according to a cost function. Besides, the time-optimal velocity planning problem (VPP) along a given path, or Time-Optimal Path Parametrization (TOPP), has also been studied over decades [87, 88]. In this section, we give a brief overview of prior work on these planning problems that has a real-time capability to be applied for the problem in dynamic environments.

2.2.4.1 Path Planning

Sampling-based Path Planning – Both RRT [84] and PRM [54] have successors that has real-time capability. For example, RT-RRT* is a tree rewiring technique for RRT which keeps removing nodes and edges that collide with dynamic obstacles and adding new nodes and edges [89]. G-Planner is a RRT-based technique to utilize GPU's parallel computation mechanism to accelerate configuration node sampling and collision checking [14, 90]. Parallel Poisson RRT also exploits GPU for tree expansion, nearest neighbor search and collision checking [91]. On the other hand, Dynamic Roadmap (DRM) algorithm is proposed built upon PRM [92]. DRM aims to reduce the collision checking time by caching robot's occupancy information into every edge of PRM's roadmap. The cached information stores voxels that are swept by the robot moving along the edges. During the planning process, the algorithm checks the intersection between the swept voxels and the voxels of the environment to determine the edge is collision-free and available or not. To reduce the memory consumption by the cached collision information in DRM, a hierarchical data structure is proposed in [93], which also contributes to a reduced planning time. Murray et al. [13] employ FPGA for DRM by encoding the occupancy data onto FPGA and parallelizing the collision checking on a chip. This enables the planner to solve the motion planning query within 1 ms. A more recent approach, Dadu-P, has a compact edge representation based on octree. This allows

the planner to reconfigure the accelerator at runtime and handle a large roadmap in a dynamic environment [94, 95].

Although these methods successfully and dramatically accelerate the sampling-based motion planning process, they produce a piecewise-linear path. The lack of smoothness makes the resulting trajectories inefficient and less human-friendly. In particular, in the context of human-robot collaboration, smooth trajectories indeed appear more predictable and acceptable to humans [9]. Therefore, they require a rapid trajectory smoother that makes the trajectory both feasible and efficient for a real robot.

Shortcutting-based Path Smoothing – The idea of path smoothing by shortcutting was first proposed by Geraerts and Overmars in [96]. Hauser et al. extend the method by introducing a parabolic trajectory representation taking into account velocity and acceleration bounds [97]. Later, Ran et al. extend this parabolic smoothing algorithm into cubic smoothing algorithm with jerk constraints[98]. Besides, Pan et al. introduce b-spline based trajectory representation and its smooth shortcutting algorithm[99]. Attempts to integrate path smoothing into sampling-based planners also exist such as RRT* and PRM* [100, 101]. For example, RRT* explores the configuration space and connects nodes with minimal cost. Eventually, RRT* generates an asymptotically optimal trajectory during planning. All of these methods can successfully compute a smooth path, but they are computationally slow because of many collision checking queries. In Chapter 3, we propose a rapid trajectory smoother to address this issue, leveraging fast clearance inference by our novel neural network. Please see Section 2.3.2 for literature review of collision detection.

Optimization-based Path Planning/Smoothing – CHOMP [85] formulates the path planning and smoothing as a quadratic problem and solve it as a sequential optimization problem with iterative linearization. This formulation differs from traditional elastic-band based approaches such as Elastic Strips [28]. Elastic Strips uses spring-mass system to represent a trajectory and performs the optimization minimizing the energy of the system, therefore it requires a collision-free initial trajectory. In contrast, CHOMP can generate a collision-free trajectory even when its initial trajectory is in collision. To address the computational burden on gradient computation, STOMP is introduced by Kalakrishnan et al. [102]. STOMP instead samples trajectories stochastically to compute the gradient, leading to more robust

convergence. While these methods can successfully compute a smooth trajectory, they are intrinsically computationally heavy and slow mainly due to non-linearity and non-convexity of the mapping between joint-space and cartesian space, and iterative gradient descent in optimization according to a given cost function. A real-time optimization-based planner using GPU [103] shows its real-time capability for dynamic obstacle avoidance by parallelizing optimization into threads that start from different random seed trajectories. However, it is uncertain that this method works as well in an environment with many obstacles due to its dependency on randomness. This method also requires models and tracking of dynamic obstacles for distance collision detection, which is not always available in reality. In contrast, ISIMP [104] plans a collision-free path within an environment represented with a pointcloud without dependency on models of obstacles. ISIMP optimizes the trajectory that is initialized by PRM*, adaptively choosing 'effective' points from the pointcloud. ISIMP may be suitable for dynamic environments since it does not require geometric model of the environment. However, it does not consider dynamic obstacles due to its slow planning speed. To achieve faster convergence, the initialization of the original trajectory is also crucial in optimization-based planners. An approach that combines a classic sampling-based planner with an optimization-based planner is proposed as ITOMP [105]. However, ITOMP also faces challenges in handling dynamic obstacles. Despite their powerful capability and flexible formulation, optimization-based planners have suffered from their heavy computational load, and their application to online planning/smoothing are currently under active research.

2.2.4.2 Velocity Planning / Time-Optimal Path Parametrization

Velocity Planning Problem (VPP), or Time-Optimal Path Parametrization (TOPP), is a problem to find the fastest velocity along a path [87, 88]. In the context of safe human-robot interaction, this problem is also known as "trajectory scaling" along with collision reaction strategy [19].

Various controllers have been proposed under Speed and Separation Monitoring (SSM) and Power and Force Limiting (PFL). The existing SSM-based methods scale a trajectory at every control iteration, formulating the problem as a linear programming optimization based on current position and velocity [106, 107]. On the other hand, prior PFL-based methods compute a maximal safe velocity by

calculating 'effective mass' or 'apparent mass' of the robot manipulator. This ensures that the accompanying force in the event of a collision remains within the limits of ISO standards and is safe enough not to injure humans [108, 109]. Recently, a method to combine SSM and PFL has been proposed which allows some force/torque when in collision to maximize the productivity and efficiency [110]. In this thesis, we focus on SSM-based controllers (see the discussion in Section 2.1). The SSM-based trajectory scaling techniques [106, 107] have a notable advantage in that they do not require the entire path to be pre-defined before execution in advance. As a result, the path is switchable at runtime, allowing for the generation of evasive or reactive motions as long as continuity is maintained. However, this advantage also comes with a downside: the method does not fully exploit the entire path information and the robot's full capability, leading to a conservative and less productive system. Additionally, these techniques do not guarantee smooth trajectories.

Aside from collaboration and safety considerations, time-optimal path parameterization(TOPP) problem has been a subject of investigation over decades. TOPP is a problem to find the fastest way for a robot to travel over a given trajectory satisfying kinodynamic constraints. One approach to tackle TOPP is by applying numerical integration to determine the maximum and minimum acceleration profile [87, 88]. This method is based on the Pontryagin Maximum Principle, which states that the optimal trajectory consists of maximum and minimum acceleration profiles in a 'bang-bang' style [111]. The algorithm finds the switching points of these max/min profiles and derives the optimal trajectory parameterization. This approach, however, has brought several difficulties [88]. Another approach is to transform TOPP into a large single convex optimization problem [112, 113]. This optimization problem is computationally demanding and slow to solve, which makes it inconvenient for online motion planning. Recently, a reachability-analysis based approach (TOPP-RA) is invented in [15]. TOPP-RA converts TOPP into a set of Linear Programming, making it well-known to be a simple, efficient, and robust solution to TOPP.

In Chapter 4, we propose a time-optimal path tracking method that enables a robot to move fastest while ensuring safety, satisfying specified constraints. This approach is built on top of TOPP-RA. Related to time-optimal path tracking, we

also review literature on batched distance computation in Section 2.3.4, which is crucial for achieving true time-optimality in path tracking.

2.3 Obstacle Perception and Processing

Real-time sensing of obstacles is another critical challenge in real-time motion planning. While specially trained masters may possess a sixth sense to perceive movement around them even with their eyes closed, robots lacking the capability to gather visual information are unable to sense their surroundings. After we briefly discuss our assumption regarding visual information in this thesis, we survey two processing components of such visual information: collision detection and distance computation.

In motion planning and smoothing, we explore a configuration space of a robot and check whether the robot collides with obstacles at every configuration the robot visits. In this process, collision detection is the bottleneck of motion planning, accounting for 90% of the entire pipeline of RRT [114]. Thus, a fast collision detector serves as a critical component for real-time motion planning, and its acceleration has been awaited in research communities. With the recent trend of multi-core processing, researchers have been motivated to parallelize the collision checking of motion planning. Specifically, our interest is in *batched* collision detection; checking collisions of a robot for multiple robot configurations simultaneously. In Section 2.3.2, we provide an overview of the recent parallelization techniques for collision checking in the context of motion planning and path smoothing.

Subsequently, we review the techniques of *batched* distance detection in Section 2.3.4. In previous research for SSM (Section 2.2.4.2), predictions of possible collisions relied on linear "extrapolation" from current states such as positions and velocities, as we will see in Section 2.3.3. These methods do not consider non-linearity of robot's motion and thus conservatively estimate its potential collisions, which causes a conservative and non-optimal behavior in collaborative robots. Instead of predicting motions, we propose computing distances discretely along the path and utilizing this information to identify the fast and safe way to traverse the path. Here, we develop an interest in *batched* distance computation. Although less focused on in previous research compared to collision detection, batched, fast, and

accurate distance computation plays a crucial role in achieving time-optimal path tracking. It enables us to anticipate possible unsafe collisions and properly control the robot's velocity.

2.3.1 Sensing Devices

In this section, we briefly overview sensing hardwares used in collaborative robotics and describe our assumption for the sensing devices in this thesis.

Early research on obstacle perception was done locally around a robot, by mounting proximity sensors such as IR sensors, ultrasonic sensors, or laser scanners on a robot arm [115]. These sensors are useful for highly responsive control [116] and may help address occlusion problems that arise with fixed cameras [117, 118]. However, these sensors do not cover the entire environment, restricting the robot's capability. Additionally, the availability of commercial ready-to-use proximity sensors in the market is limited, leading to a broader adoption of fixed sensors in both research and industrial applications nowadays. In this thesis, we assume the use of fixed sensors in the environment.

There are many options available for fixed sensors, such as motion capture, laser range sensors, cameras, and RGB-D sensors. Motion capture is widely used in research of human-robot collaboration [49, 119, 120] and one of the best options to choose since it is accurate, fast, and robust. However, its primary disadvantage is that we have to attach markers to every single object and all the joints if articulated, which may not always be feasible and cost-effective in real-world scenarios. For example, in an automotive factory setting, a human worker might handle numerous parts in a large assembly engine using various tools such as wired screwdrivers and wrenches. It is impractical to attach markers to all of them. Laser range sensors are also widely used to ensure workspace safety. Traditionally, cost-effective 1D or 2D laser sensors are often employed as a 'virtual wall' to detect the approach of a human into the robot's workspace [121]. With the emergence of Kinect [122], 3D laser sensors and RGB-D sensors start to be broadly used for obstacle perception [123] and human-robot collaboration [46, 123]. Subsequently, depth image and/or 3D pointcloud captured by Kinect have enabled human skeleton tracking [106, 124]. However, for this thesis, we do not rely on skeleton tracking

due to its lack of 100% reliability, particularly when a human is occluded or partially in the field of view [125]. It is worth noting that skeleton tracking is still an active area of research.

In summary, our thesis holds an assumption that tracked models of obstacles and human skeletons are not available and we can only rely on a pointcloud of the environment obtained from fixed camera(s). The use of motion capture or detailed human skeleton tracking is not part of our approach. We refer readers to [115] for a detailed review.

2.3.2 Batched Collision Detection

Collision detection between 3D models has been extensively studied and finds many applications such as computer graphics, CAD, gaming, and robotics [126]. To enhance the speed of collision detection, researchers have explored parallelization techniques. Their efforts have been mainly concentrated on accelerating a *single* collision detection query in a scene [127–129], parallelized collision detection between multiple objects/agents in a *single* scene [130, 131], or handling complex objects like cloth and deformable objects [132–134]. In contrast, our interest lies in *batched* collision detection, which involves checking collisions between a robot and obstacles with *multiple* robot configurations simultaneously.

The first contribution for batched collision detection can be found in [114]. In this work, they identify that collision detection is the bottleneck of RRT and propose a parallelized 2D collision detection algorithm on GPU for a simple multi-link robotic manipulator. While their model may be too simplified for real-world applications, they experimentally show the effectiveness of batched collision detection in motion planning. Their core idea remains valid and can be extended to more complex robots.

The main difficulty in collision detection arises from the non-linearity of the mapping between joint space and Cartesian space. Recent approaches to tackle this problem involve pre-computing collision information in a discretized manner and caching it for runtime efficiency. For example, Hermann et al. [123] sample points from the surfaces of robot links, transform them for each waypoint of a trajectory, and pre-compute occupied voxels swept by the robot as it traverses the trajectory.

At runtime, collision detection is performed by examining whether pre-computed voxels are occupied by obstacles or not. Another approach by Murray et al. [13] pre-computes the swept voxels by a robot traversing each edge of a roadmap, and encodes them onto an FPGA device. They similarly check collisions for each edge of the roadmap through bit-wise AND operations between the pre-computed voxels and the voxels actually occupied by obstacles on the chip. This allows the PRM-based motion planner to plan a path within 1 ms . Dadu-CD [135] adopts a more compact occupancy data representation based on octree, allowing the planner to reconfigure the accelerator at runtime and to handle a large roadmap in a dynamic environment with lower energy consumption. These approaches are well-suited for real-time motion planning. However, they can check collisions only for pre-computed configurations and cannot handle collisions for unseen configurations.

With recent advancements in machine learning techniques, machine learning has been applied to collision detection through the estimation of batched collisions/-clearances between a robot to its environment. Notable researches include CN-RRT [136] and Fastron/DiffCo [137, 138]. ClearanceNet in CN-RRT is a neural network that takes joint values as its input and estimates the distance between a robot and its environment, with the output distance(s) being thresholded to determine collision status. Fastron, on the other hand, is a binary estimator of collision/free-space from input joint values. These approaches propose an implicit function that maps from state variables to collision information. While they successfully integrate clearance/collision estimation into random-sampling based motion planning (CN-RRT) and optimization-based motion planning (DiffCo), CN-RRT lacks the adaptability to handle dynamic obstacles adaptively, and DiffCo requires 'active learning' at runtime to manage dynamic obstacles. This makes it hard to guarantee the reliability of the neural model. Moreover, DiffCo's optimization-based smoothing takes more than 2 seconds in total reportedly, which is not suitable for online motion planning. The most recent learning-based approach, GraphDistNet [139], builds on Graph Neural Network [140] and achieves better precision of distance-estimation compared to CN-RRT and DiffCo. However it is limited to a 2-dimensional representation and has the same limitation with CN-RRT and Fastron in terms of reliability under a dynamic setting. In Chapter 3, we propose a rapid path smoother method that utilizes fast clearance inference by our neural network, capable of smoothing a trajectory within 0.2–0.3 seconds including exact

geometric collision checking for safety assurance. Our approach can handle dynamic obstacles without requiring any runtime modification to the neural network model or other parameters.

2.3.3 Human/Robot Motion Prediction

Motion prediction is a technique to foresee the future motion of an object based on its current state, including positions and velocities, taking into account its capabilities such as accelerations and torques. This has the potential to improve the safety and productivity of robots operating in a dynamic environment.

Human's reachable space prediction is one direction of such techniques, as explored in various studies [25, 107, 120]. Given a kinematic model of a human skeleton and positional/velocity/acceleration limits of human joints, the reachable space can be computed as a set of swept volumes of convex human links. Motion prediction is beneficial for planning faster motion because it enlarges the robot's configuration space which is expected to be collision-free. However, this approach relies on precise tracking of the human skeleton's position and velocity, which may not necessarily be reliable and is still under active research. Considering that the maximum acceleration of hand motion can be up to 6 m/s^2 [141], human motion prediction is intrinsically challenging with external, cost-effective sensors. For this reason, we do not rely on human motion prediction in this thesis. We, instead, assume that only the positions of obstacles and their maximum velocities are given and that the obstacles can change their direction quickly while maintaining their maximum speed. Please note that human motion prediction can be easily integrated with our method proposed in Chapter 4 and 5.

Instead of predicting human behavior, the robot's reachable space has also been considered for collision avoidance and speed control. The idea of robot's reachable space can be traced back to Velocity Obstacles(VO) [142] and Inevitable Collision States(ICS) [143]. These methods consider collision detection in velocity space and are often used in 2D mobile robots for adaptive control [144] and local planning [145]. An extension of VO to a planar two-link manipulator can be found in [146]. However, due to the non-linearity of the mapping between the robot's configuration space and Cartesian space, VO/ICS are not directly applicable to

articulated robots. Therefore, current SSM-based controllers, introduced in Section 2.2.4.2, are based on linear approximation of collision prediction from current distances and link velocities of the robot [106, 107, 110].

In relation to motion prediction, dynamic separation distance has been introduced to mitigate the conservativeness of the SSM-based methods (Section 2.2.4.2). In SSM, the distance between a robot and a human must be larger than the separation distance while the robot is moving, and it can be dynamically adjusted according to the situation. For example, Vicentini et al. [147] compute a trajectory-dependent separation distance dynamically based on link velocities to improve the robot's reaction time. Similarly, Glogowski et al. [148] calculate distances from the entire kinematic chain of a robot to a human, and adaptively control the robot's speed accordingly. In the commercial realm, Veo Robotics Inc. offers a collaborative robot controller FreeMove®that 'calculates possible future robot positions and signals the robot to stop by sending safety outputs that indicate if the robot is closer to a human than the Protective Separation Distance' [149]. However, as these methods are based on motion prediction from current instant information, they do not consider the non-linearity of the robot's motion and conservatively compute the separation distance, which still results in overly cautious and non-optimal behavior.

2.3.4 Batched Distance Computation

2.3.4.1 Model-based Distance Computation

In collision detection and distance computation between 3D models, hierarchal data structure, or 'broad-phase structure', is commonly applied to filter out object pairs that are far away and dramatically accelerates the collision/distance queries. Examples of such data structures include AABB Tree, OCTree, and Inner Sphere Tree [16, 150, 151]. However, these data structures are optimized for CPU and lack batch-processing capabilities, hence they do not have a sufficient throughput for real-time safety control. For instance, the distance query with an octree for 1000 configurations will take 39.1 ms according to [150] (excluding the octree construction time), which remains slow for real-time control. Another challenge is that the throughput depends on the positions of robots and obstacles, which

makes it difficult to ensure its constant execution time at the time of deployment of the system. In contrast, our approach proposed in Chapter 5 does not depend on runtime-varying settings except for the number of configurations.

Simplification of robot/human models with geometric primitives such as spheres and capsules is commonly used for distance computation in motion planning and safe robotic operation [85, 124, 152, 153]. However, evaluating distances for multiple configurations in a batched manner using primitives other than spheres is actually slow. In fact, in our preliminary experiment, distance computation between a 6 DoF robot simplified with 7 capsules and (only) 3000 points for 500 configurations took about 70 ms even on GPU, which is not feasible for real-time control. This is primarily because the projection of points onto the axis of capsules requires a time-consuming (batched) matrix multiplication at runtime.

2.3.4.2 Signed Distance Fields (SDF)

In unknown environments, prior knowledge about obstacles such as their shape, size, and position is not always accessible. Therefore, Signed Distance Fields(SDF) or Un-signed Distance Fields(USDF) are often employed as these do not necessarily require prior knowledge of obstacles [154]. There can be two approaches to computing a distance using SDF; 1. compute the SDF of an *environment* and retrieve it for the robot, or 2. compute the SDF of a *robot* and retrieve it for the environment.

In optimization-based trajectory planning, the (U)SDF of an *environment* is often used because it offers a gradient to push the trajectory away from obstacles [85, 155, 156]. There are a number of methods for SDF construction; some stem from a context of SLAM [157–159] and others from the domain of machine learning/NeRF [160]. However, most existing methods assume a *static* environment and incrementally construct a scene since SDF reconstruction requires inherently time-consuming data propagation. To the best of our knowledge, no previous work satisfies all the requirements for our safe-control application: 'batched', 'real-time', and 'precise'.

One of the promising work is [161] which builds on [162, 163]. This method computes the SDFs of an environment from an incoming sensory pointcloud in parallel on GPU using a Parallel Banding Algorithm [164]. The distance data

for the voxels occupied by the robot is then retrieved. In their demonstration, they showcase online motion planning of a mobile manipulator platform. However, the computation time of this method depends on the size of the environment due to the data propagation and is not suitable, especially for large environments. Most importantly, their reported SDF construction time is 17.5 ± 0.4 ms for 5cm resolution and 36.2 ± 8.3 ms for 2.5 cm resolution, which is not fast enough for real-time control (Note that the 'SDF computation time' does not include the time for distance queries yet).

Another interesting work is ReDSDF [165], a machine learning-based SDF estimator that employs a neural network that takes query points and poses as inputs and outputs distances for each of them. While its estimation accuracy is sufficient for safety-critical use cases, its architecture is not suitable for real-time safety control due to the following reasons: For robot's SDFs generation: 1. ReDSDF requires retraining of a neural network for any change in a robot, and 2. the neural network needs to be evaluated for each waypoint along a trajectory, repeatedly feeding the same query points. These requirements make ReDSDF unsuitable. For environment's SDFs construction: 1. ReDSDF requires a model that is trained for each individual obstacle, and 2. each obstacle needs to be tracked in some way; these are not realistic for industrial applications.

Some previous methods employ link-local SDFs of a *robot* for distance computation in motion planning or collision avoidance [166, 167]. In these approaches, the pointcloud is transformed into every link coordinate, and the distance is then obtained from link-local SDFs, resulting in a computational complexity of $O(DM)$ where D is the DoF of a robot and M is the number of pointcloud. This computation heavily depends on the number of points and its batch processing is often insufficiently fast for real-time robot control. In contrast, our proposed method in Chapter 5 performs the transformation of the link-local SDFs onto the coordinates of the environment beforehand, eliminating the need for pointcloud transformations (Fig. 2.2). This leads to a faster distance retrieval in the real-time distance computation phase.

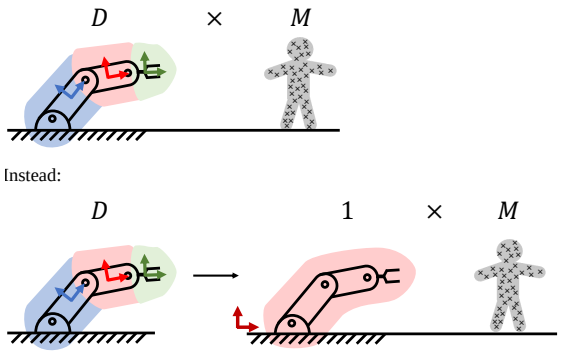


FIGURE 2.2: A common way to compute a distance between pointcloud and SDFs requires pointcloud transformations for each link coordinate. Instead, we transform and merge the link-local SDFs into the global coordinates in a pre-processing stage, and then evaluate it to obtain distances at runtime.

2.4 Summary

In this chapter, we have reviewed the literature on motion planning and control and obstacle perception under dynamic environments for safe robot collaboration. We discussed the strengths and weaknesses of current technologies from an industrial perspective, identifying the research gap in the field. Specifically, we focus on the problem of path smoothing and time-optimal path tracking in real-time motion planning, and their interrelated problems of batched collision detection and distance computation. To ensure the safety of human operators, we make the assumption that a robot must stop before it collides with a human in accordance with SSM framework. Furthermore, we consider a common scenario where obstacle models are not available and the only available data is a pointcloud of the environment captured from fixed cameras. From the next chapter, we will discuss our contributions to these problems. We provide a rapid trajectory smoother for TPP in Chapter 3, the missing piece in real-time motion planning, and a time-optimal path tracker for VPP in Chapter 4, which guarantees the safety of human operators conforming to ISO standards. Additionally, we propose a batched distance computation method for truly time-optimal path tracking in Chapter 5.

Chapter 3

Rapid Trajectory Smoothing with Neural Nets

3.1 Introduction

In order to safely and efficiently collaborate with humans, robots need the ability to alter their motions quickly to react to sudden changes in the environment, such as an obstacle appearing across a planned trajectory. In most industrial applications, one would stop the robot upon the detection of obstacles in the robot's reach space. However, such a solution is inefficient and precludes true human-robot collaboration, where humans and robots are to share a common workspace.

Recently, Realtime Motion Planning (RMP) has been proposed to enable true human-robot collaboration: obstacles are detected in real time through a vision system, new trajectories are planned with respect to the current positions of the obstacles, and immediately executed on the robot. RMP requires extremely fast computation in the Vision–Motion Planning–Execution loop. In particular, several techniques have been proposed for the Motion Planning component, relying on the parallelization of sampling-based algorithms [54, 84] on dedicated hardware, such as GPU [14, 90] or FPGA [13].

Sampling-based motion planners typically output jerky trajectories and therefore almost always require a smoothing post-processing step (see e.g. [168] for a detailed review). To our knowledge, existing real-time motion planners lack this

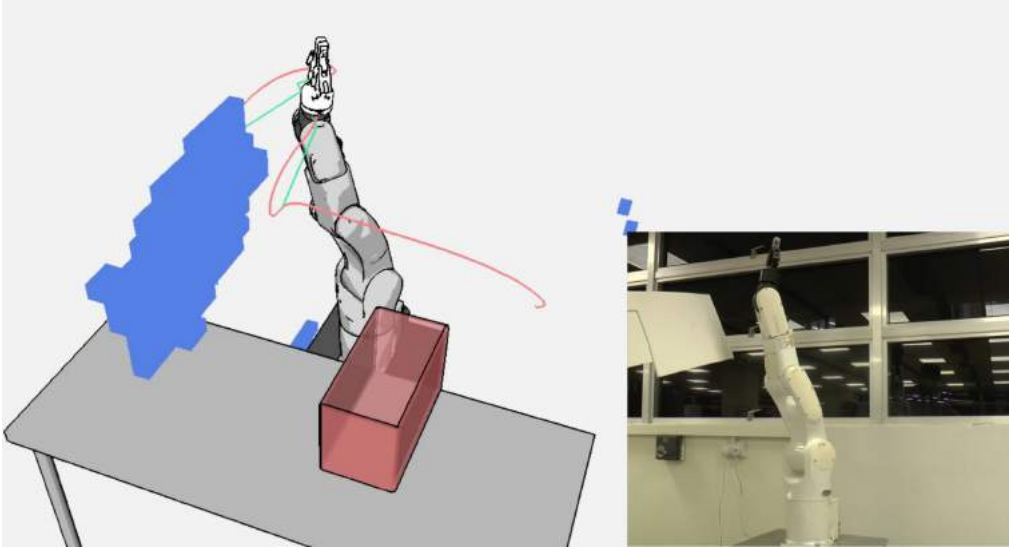


FIGURE 3.1: A robot avoids a dynamic obstacle by realtime trajectory replanning and smoothing with the proposed smoother. Green: jerky trajectory planned by realtime PRM. Red: trajectory after smoothing. See the video of the experiment at `https://youtu.be/XQFEmFyUaj8`.

step [1], presumably because of the large computation time associated with trajectory smoothing. Indeed, to obtain an acceptable trajectory quality, *smoothing time is comparable, if not longer than initial path planning time* [168]. As a result, while the motions produced by Realtime Motion Planning enable safely adapting to sudden changes in the environment, the lack of smoothness makes them inefficient and less human-friendly. In the particular context of human-robot collaboration, smooth trajectories indeed appear more predictable and agreeable to humans (see e.g. [9] for a discussion).

Here we propose a Rapid Trajectory Smoother based on the shortcutting technique [96, 97] to address this issue. Leveraging fast clearance inference by a novel neural network, the proposed method is able to consistently smooth the trajectories of a 6-DOF industrial robot arm within 200 ms on a commercial GPU, which is 2 to 3 times faster than state-of-the-art smoothers. Combined with even rudimentary, in-house, implementations of a vision pipeline and a sampling-based motion planner, we were able to achieve a 300 ms cycle time, which is sufficient for real-time

[1] Video: Yaskawa Motoman Demo by Realtime Robotics on Vimeo `https://vimeo.com/359773568`. Note the jerky transitions, for example at time stamps 24s, 30s, and 35s.

performance (Fig. 3.1). Note again here that smoothing is indeed the bottleneck, as the initial planning time was only ~ 40 ms.

The chapter is organized as follows. In Section 3.2, we present our trajectory smoothing pipeline and the structure of the neural network model for fast clearance inference. In Section 3.3, we evaluate the pipeline in two sets of experiments. First, we evaluate the clearance inference accuracy of the neural network. Second, we integrate the smoother into a full-fledged Vision–Planning–Execution loop, and demonstrate a rapid, smooth, performance of a physical industrial robot subject to dynamic obstacles. Finally, we discuss the advantages and limitations of our approach and conclude with some directions for future work (Section 3.4).

3.2 Realtime Trajectory Smoothing

3.2.1 Overview

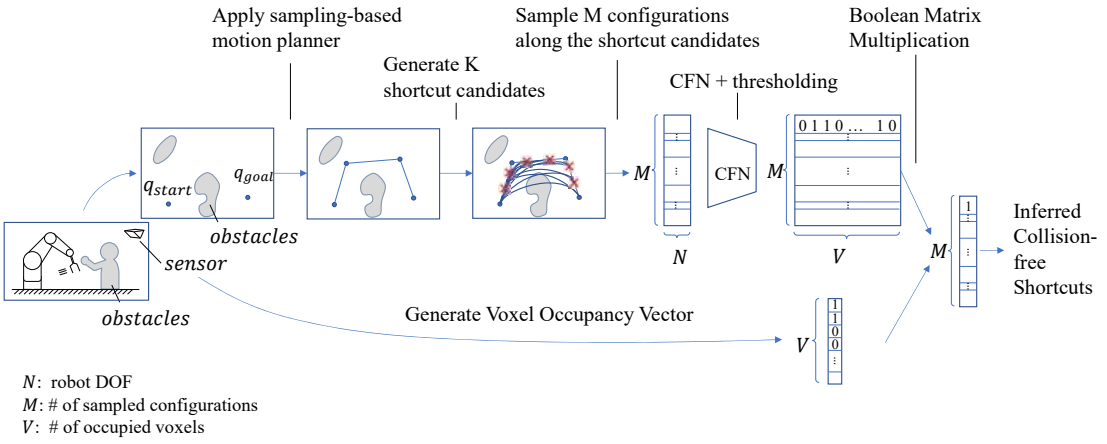


FIGURE 3.2: A pipeline of trajectory collision estimation in NN-accelerated trajectory smoother. See Section 3.2 for detail.

The pipeline of our NN-based trajectory smoother is illustrated in Fig. 3.2. Consider a N-DOF robot. Given a piecewise linear path (blue lines) in the configuration space, typically outputted by a sampling-based planner, our smoother samples c configurations at regular time-intervals (red X marks), and computes parabolic shortcut trajectories between every pair of sampled configurations. We have $(c+2)$ configurations in total on a given path which consists of the c sampled waypoints

plus a starting configuration q_{start} and a goal configuration q_{goal}. Let them be $q_0 ... q_{c+1}$, and a shortcut from q_i to q_j be $S(q_i, q_j)$. We have $K = \frac{(c+2)(c+1)}{2}$ shortcut candidates, namely, $S(q_0, q_1)$, $S(q_0, q_2)$, ... $S(q_0, q_{c+1})$, $S(q_1, q_2)$, $S(q_1, q_3)$ ... $S(q_1, q_{c+1})$, ... $S(q_c, q_{c+1})$.

Given a shortcut $S(q_i, q_j)$, we sample m_{ij} configurations at regular intervals along the shortcut. Next, we stack the $M = \sum_{ij} m_{ij}$ configurations into one single $M \times N$ matrix. This matrix is then fed into the "Clearance Field Neural Network" (CFN, see details in Section 3.2.2) for *batch processing*. Assume that the spatial workspace is discretized into V voxels, the CFN returns a matrix of size $M \times V$ containing the inferred clearances from the robot placed at every sampled configuration to every voxel of the discretized spatial workspace. Next, we perform thresholding to obtain the $M \times V$ Inferred Collision Matrix: a configuration q is considered as in collision with a voxel if the clearance from the robot placed at q to the voxel is smaller than a given threshold.

In parallel, from the real-time pointcloud captured by the vision system, we generate the $V \times 1$ Voxel Occupancy Vector: an element of this vector is 1 if the corresponding voxel is occupied by an obstacle, 0 if not.

Next, we perform a boolean matrix multiplication of the $M \times V$ Inferred Collision Matrix by the $V \times 1$ Voxel Occupancy Vector to obtain a $M \times 1$ vector that gives the inferred collision status for all the sampled configurations, which in turn yields the inferred collision status for all the K shortcut candidates (a shortcut candidate is in collision if any of its sampled configurations is in collision).

Finally, we run the Dijkstra algorithm to find the shortest trajectory consisting of *inferred collision-free* shortcuts. We then check the *actual* collision status of the obtained shortcutted trajectory by a geometric collision checker. If the inference is exact, the shortcutted trajectory should be collision-free and selected. If not, we re-run Dijkstra until an actually collision-free trajectory is found. In practice, owing to the good inference quality, we observed that the shortcutted trajectory returned by the first Dijkstra call is actually collision-free 86.1% of the time.

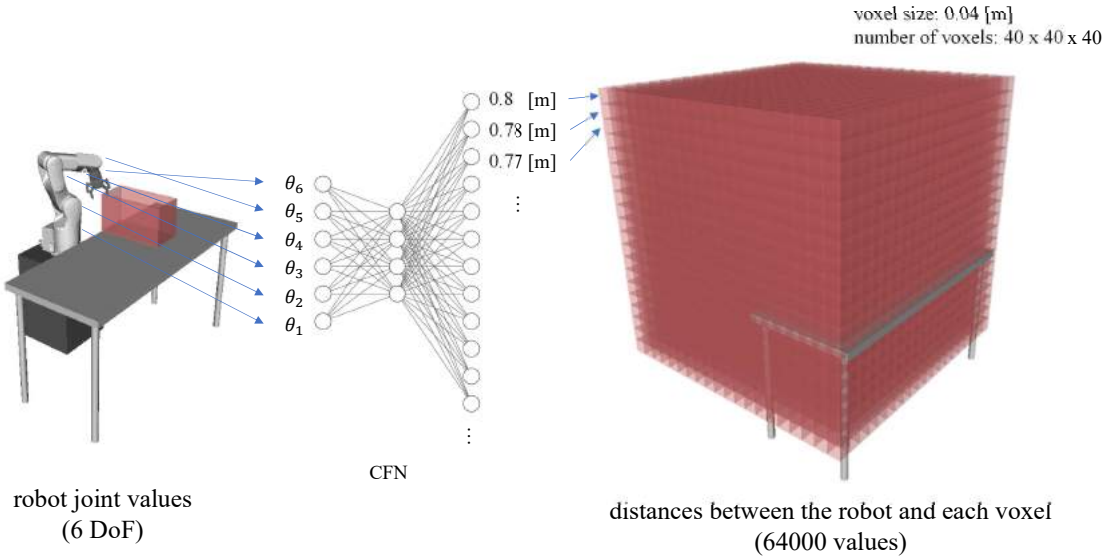


FIGURE 3.3: The environment is discretized into V voxels. A voxel's clearance is the clearance between the voxel and the robot surface. A Clearance Field is a $V \times 1$ vector that contains all the clearances of the V voxels. As the Clearance Field depends on the robot configuration, a Clearance Field Network learns the mapping from a configuration q to its ClearanceField(q).

3.2.2 Clearance Field Network (CFN)

We formally define the clearance of a voxel as the signed-distance between the voxel and the robot surface. Note that the clearance can be negative if the voxel is "inside" the robot. The Clearance Field is then defined as a $V \times 1$ vector that contains all the clearances of the V voxels. Observe that the Clearance Field depends on the robot configuration. A Clearance Field Network is a Neural Network that learns the mapping (see Fig. 3.3 as well):

$$\begin{aligned} \mathbb{R}^N &\rightarrow \mathbb{R}^V \\ q &\mapsto \text{ClearanceField}(q). \end{aligned}$$

To learn this mapping, we follow a supervised learning approach: offline, we generate a large number of random configurations. For each configuration q, we use a geometric collision-checker (which provides clearance data, such as FCL [16]) to calculate the clearance at every voxel, constructing thereby ClearanceField(q). At run time, given a new, possibly unseen q_{new}, one can quickly infer ClearanceField(q_{new}).

The architecture of the proposed Clearance Field Network is shown in Fig. 3.4. The "sin, cos kernel" converts joint values q to:

$$\begin{aligned}\ker(q) = [&\sin(2^0\pi q), \cos(2^0\pi q), ..., \\ &\sin(2^{L-1}\pi q), \cos(2^{L-1}\pi q)]\end{aligned} \tag{3.1}$$

inspired by NeRF's positional encoding [169]. This is intended to increase the frequency of input values and allow the neural network only to learn low-frequency features.

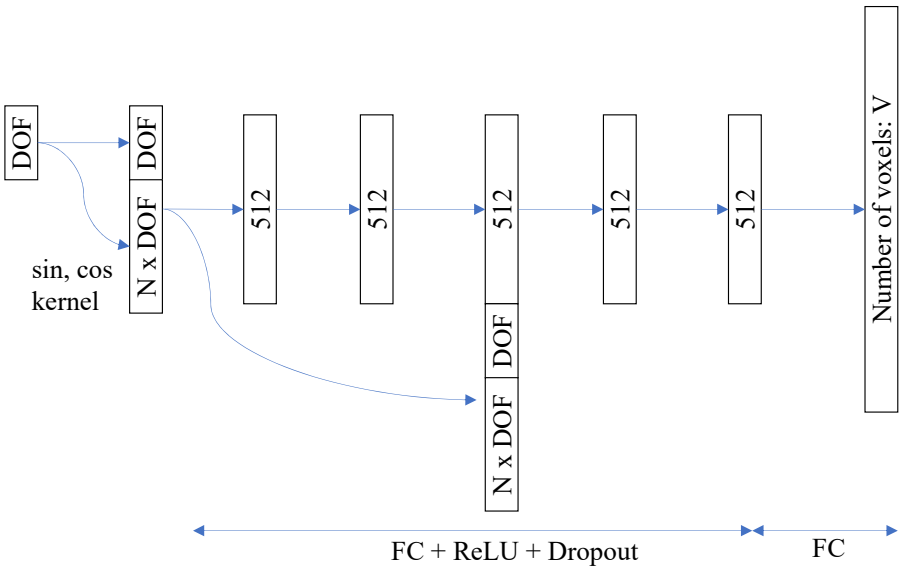


FIGURE 3.4: The architecture of Clearance Field Network: Kernel function inputs a high-frequency values into a neural network using sine and cosine. The middle layers are composed of Fully-Connected, ReLU and DropOut with one skip connection.

The advantage of this method is that it can handle dynamic obstacles. Previously, [136] needs to feed the information of dynamic obstacles into the neural network. However, since the dimension of input of the neural network should be static and fixed, it is needed to convert the variable size of information about dynamic obstacles into a static-sized feature vector, and feed it into the neural network. In contrast, in our approach, the dynamic obstacles already exist in the form of occupied voxels, whose number is fixed. Our approach can thus apply to any number/shape of dynamic obstacles and the computation can be easily parallelized on GPU.

The reason why we propose to learn clearances instead of directly collision status is that neural networks are better at approximating continuous functions, and

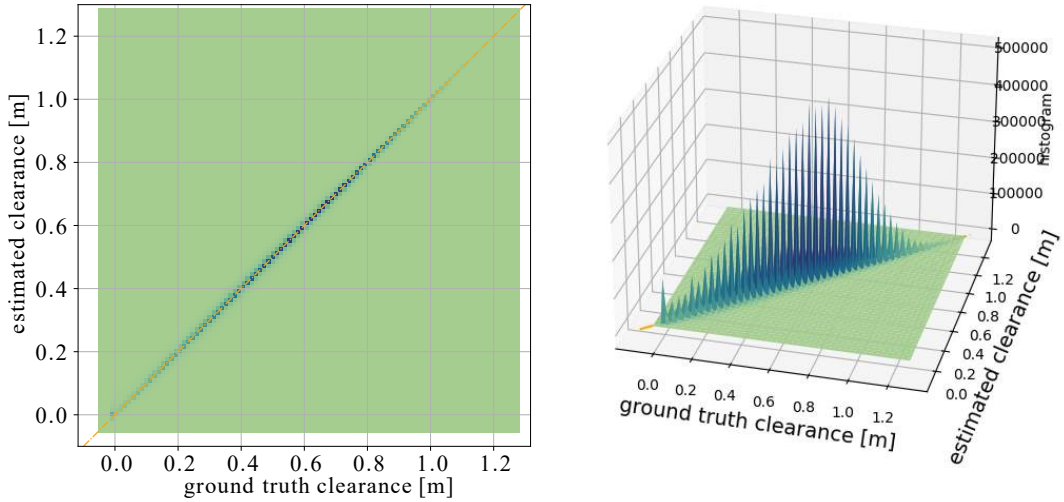
clearance is a Lipschitz-continuous function of the environment[136], while collision status is a discrete function.

3.3 Experiments and Results

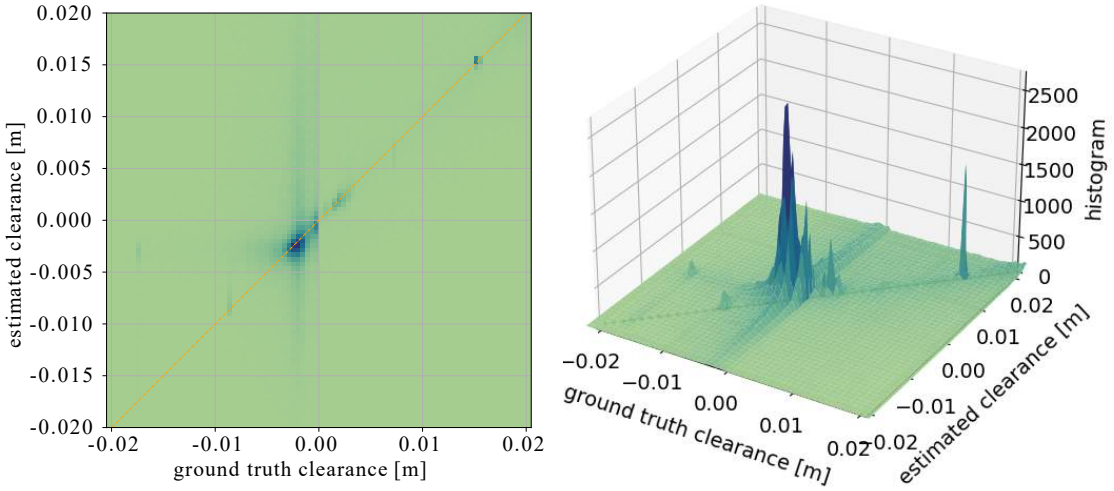
3.3.1 Performance of CFN

First, we examine the clearance field network. We train our neural network with 52,000 joint values and their corresponding clearances using a batch size of 50, validate the training process with 16,000 validation data, and test the trained network with 12,000 test data. During training, we use L1 loss and Adam optimizer [170] with a learning rate of 1×10^{-3}. We set $L = 3$ for positional encoding in this experiment. The total data generation takes 2.7 days using 16 CPU cores. The optimization takes 300 iterations (about 1 hour) to converge. Note that the above data generation and optimization steps need to be done only once for a given robot model (without obstacles).The obstacles are handled by the fast *inference* step at execution time.

We show colored 2D/3D histograms of clearances estimation in Fig. 3.5. False negatives (i.e. robot position in collision being classified as free) lead to an invalid trajectory and more time for shortcutting, whereas false positives (i.e. collision-free robot position being classified as in collision) lead to conservative shortcutting and a longer trajectory. This trade-off can be managed by a clearance threshold. Precision ($\frac{\text{Estimated as incollision}}{\text{Actually incollision}}$) is 85.3% and 90.9% when we set a threshold as 20 [mm] and 30 [mm] respectively, which are sufficient to infer the collision status of trajectories. By setting a large threshold, we can reduce the probability of obtaining a geometrically in-collision shortcutted trajectory while it may lose a shorter trajectory and vice versa. We use 20 [mm] in our later experiment. Note that safety is always guaranteed because the smoothed trajectory is verified by a geometric collision checker at the end of our pipeline.



(A) All test results



(B) Zoomed in the origin

FIGURE 3.5: Colored histograms of Clearance Field Network estimation. Upper: a 2D histogram using all 12,000 test data (left) and its 3D mesh plotting (right). Lower: zoomed in the origin with higher resolution (left) and its 3D mesh plotting (right). An orange dotted line represents a 45 degree line for a reference.

3.3.2 Integration of CFN into a motion planning pipeline

Secondly, we compare our proposed smoother using the trained neural network with OpenRAVE's state-of-the-art parabolic smoother [171] which originates from [97] and is updated with recent theoretical advancement[172]. In this experiment, given a piecewise linear trajectory from a start to a goal (from 'config1' to 'config2' in 'Scene1', and from 'config3' to 'config4' in 'Scene2' shown in Fig. 3.6), smoothers smooth the trajectory and we measure their computation time and the duration of its smoothed trajectory under different configurations, changing the maximum number of iterations for the existing method, and the number of sampled waypoints for our proposed method. The clearance threshold for our collision estimation is set to 20 [mm]. We sub-sampled shortcuts at 0.04 seconds interval. Our code is based on OpenRAVE[173] and we use FCL[16] as a collision checking library. All the experiments are done on a single machine, on which Intel® Xeon® W-2145 and GeForce GTX 1080 Ti are mounted for CPU and GPU.

In Fig. 3.7, we plot ratio of smoothed trajectory duration to unsmoothed initial trajectory duration v.s. computation time for each method, where a shorter trajectory and a faster computation time (the left, bottom side of the figure) is preferable.

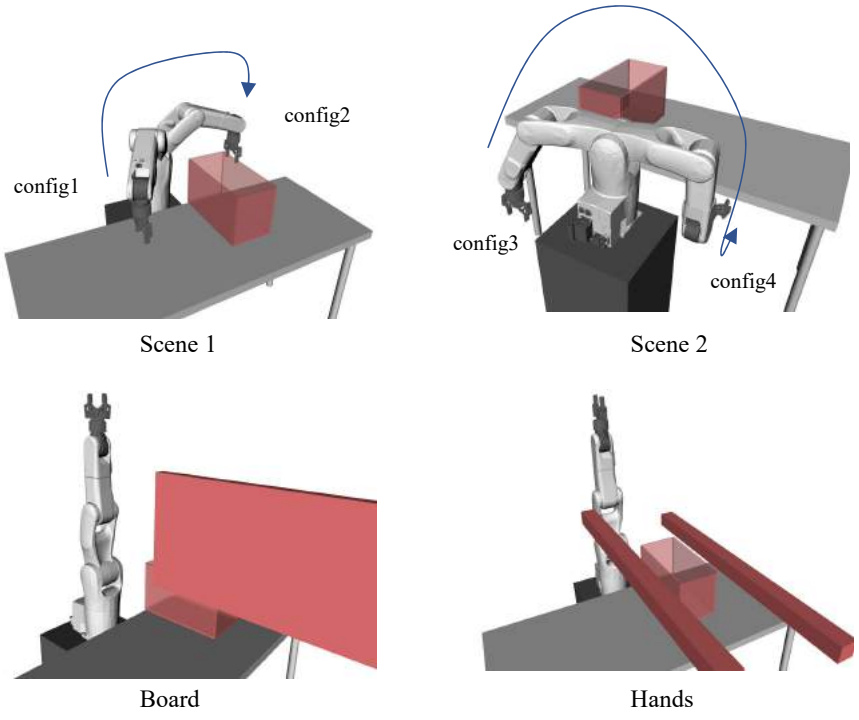


FIGURE 3.6: Scenes and Obstacles for trajectory smoothing computation time comparison experiments

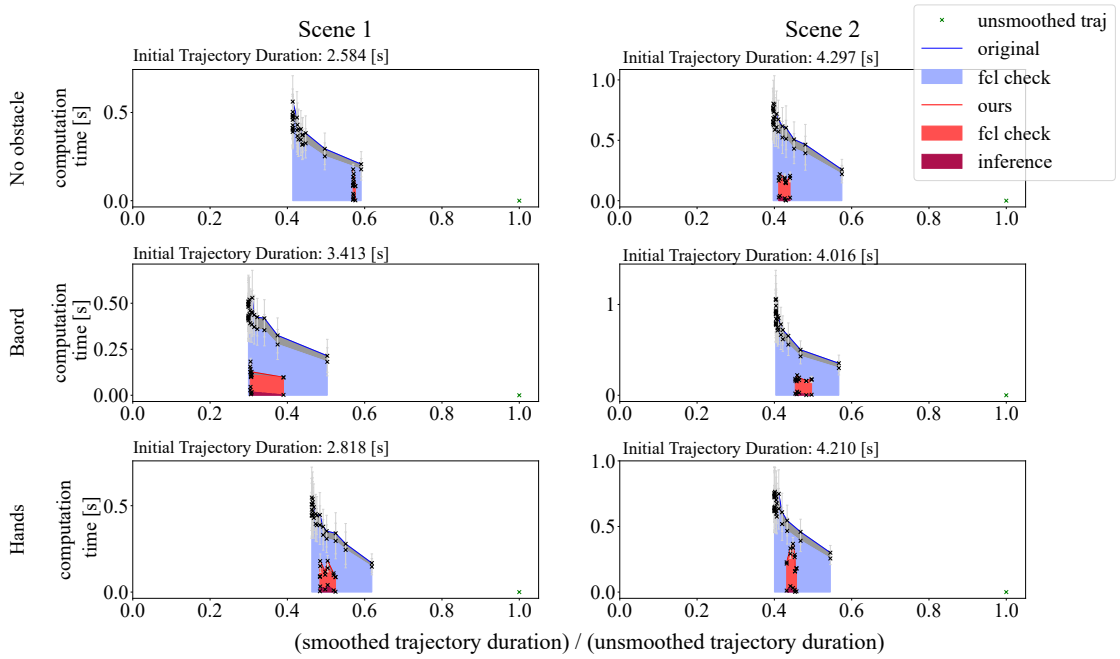


FIGURE 3.7: Reduction Ratio of Smoothed Trajectory Duration to Unsmoothed Trajectory Duration to Shortcutting Computation Time [s]. 'original' is OpenRAVE's state-of-the-art parabolic smoother [97, 171]. 'fcl check' represents geometric collision checking by FCL[16] and 'inference' represents collision inference time. See Section 3.3.2 for detail.

In the existing method, the smoothing time increases as we increase the number of max iterations generating shorter trajectory (blue line), whereas in our method, the smoothing time does not increase constantly as we increase the number of sampled configurations (red line). The result shows that the computation speed of our proposed method is generally 2–3x faster than that of the existing smoother to generate trajectories of the same length, and can generate a much shorter trajectory within the same computation time.

In some cases, e.g. Scene2 with Hands in Fig. 3.7, the time of geometric collision checking increases to find another collision-free trajectory when false-negative collision detection happens, i.e., it can not correctly estimate collision-free trajectories. Statistically, the first inferred collision-free candidate is actually collision-free in 86.1% of 36 cases, and the second, third, and fourth candidates are selected in 2.7% (once), 5.6% (twice) and 2.7% (once) of the cases consecutively.

In terms of optimality, our smoother cannot find an optimal trajectory as we see

in Fig. 3.7. There are several reasons: First, we do not have multi-staged waypoint sampling as it is done in [97], therefore the robot does not accelerate well along the smoothed trajectory. Second, we convert a pointcloud into voxels which include original obstacles. This conversion adds voxel-size padding at maximum to obstacles. Third, we set a clearance threshold in collision estimation, which can also be interpreted as padding and makes the robot larger than its real size, leading to a longer trajectory. However, our focus is not on asymptotical performance but on reducing one-shot smoothing time against computation time, and this non-optimality is acceptable for our use case, i.e., realtime motion planning.

Finally, we conducted a physical robot experiment (Fig. 3.8). The robot loops between point A and point B while the experimenter randomly introduces obstacles on the robot's path. The obstacles are monitored by a Kinect v2 mounted at the side of the workspace, and the obtained pointcloud is converted into a Voxel Occupancy Vector by thresholding the number of points in every voxel at a certain threshold (50 in our experiment) to reduce the effect of sensor noise. When we compute shortcut-candidates in re-planning, the robot's velocity at the new q_0 is considered to enable a smooth transition from the current trajectory to a new re-planned trajectory. By bringing the computation time of the entire Vision–Planning–Execution loop under 300ms, our smoother enables the robot to react to fast perturbations (we observed that using OpenRAVE's state-of-the-art parabolic smoother, such real-time reaction was not possible as the robot had to stop, pause to compute a new trajectory, and restart), while being smooth (compare with Realtime Robotics' demo `https://vimeo.com/359773568`).

3.4 Summary

We have proposed a trajectory smoother by leveraging a neural network to estimate clearances and collisions between a robot and voxels in a parallelized manner. Our planner is 2–3x faster than an existing method, making real-time performance possible.

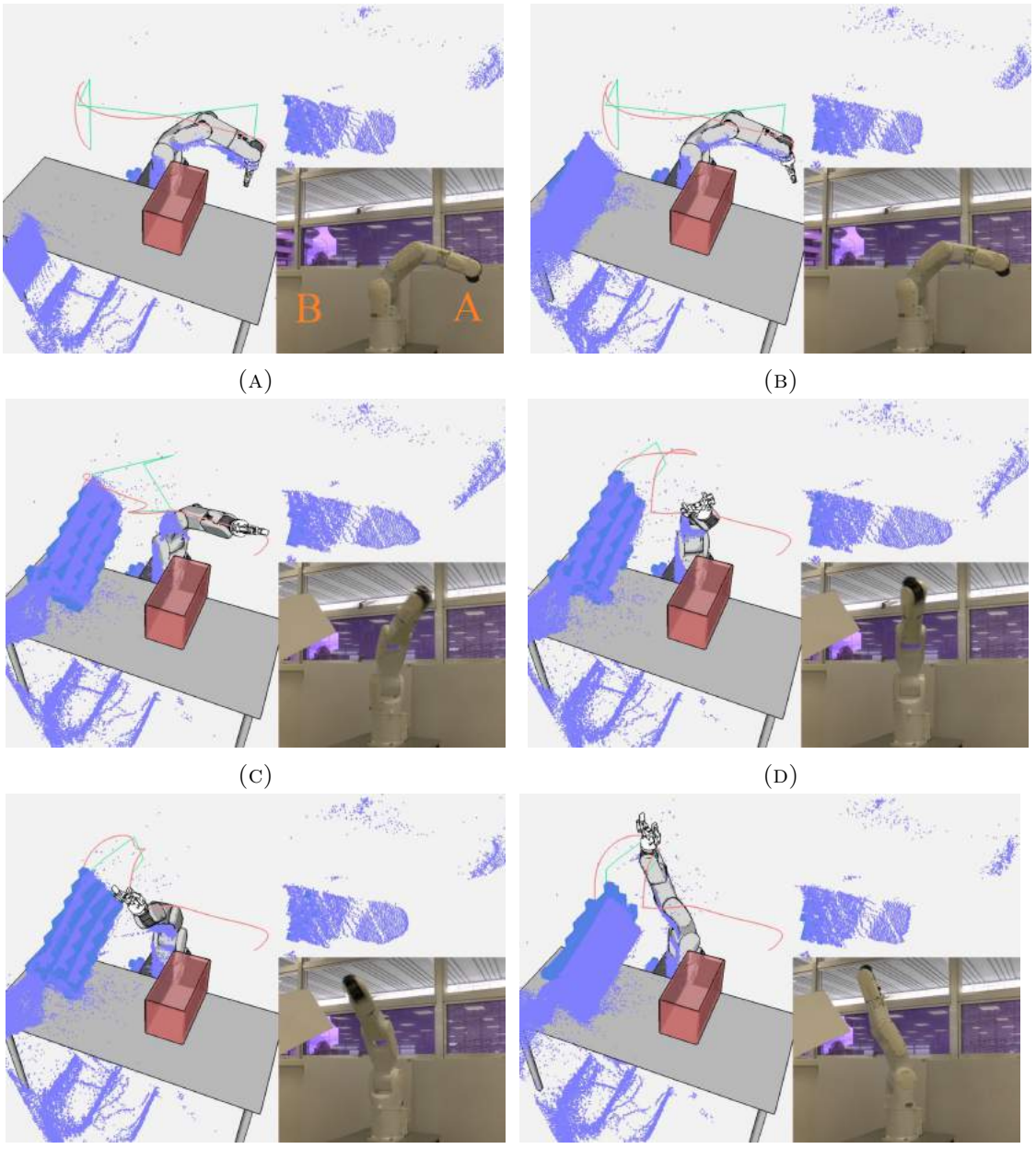
Why is our smoother faster and can generate shorter trajectory within the same amount of computation time? The reason for shorter trajectories is that our

smoother aggressively tries to connect distant waypoints. Even when the number of sampled waypoints is small, our smoother tries to connect waypoints far away from each other (one of them connects a starting point to a goal), so that the computed trajectory tends to be shorter. In contrast, with the existing method, when the number of iterations is small, there is less possibility to connect distant waypoints to have a shorter shortcut. The reason why it is faster is that NN collision estimator can *batch-evaluate* many waypoints using significantly less time than the geometric collision checker.

Currently, there are a number of limitations, which we intend to address in future work.

- Memory consumption and scalability: To find an optimal trajectory as much as possible, we need to increase the number of sampling and have a smaller voxel size, which leads to a larger batch size of input and large memory consumption. Under the configuration in Section 3.3, our neural network model itself takes about 1.5 GB of GPU memory, and we allocate 0.8 GB of GPU memory for temporary variables. This memory consumption increases in $O(c^2VL)$ where c is the number of sampled waypoints, V is the number of voxels and L is the length of the trajectory (c^2 because we have $K = \frac{(c+2)(c+1)}{2}$ shortcut candidates). Therefore we need to prepare a GPU with large memory to increase the number of sampling waypoints, decrease the resolution of voxels, and apply for a large environment.

- Planning Constraints: Our smoother assumes that collision checking is the bottleneck of trajectory smoothing, i.e., the first shortcut-candidates computation step is much faster than collision checking. If we need to take into account other constraints such as torque limits and robot hand/grasped object's orientation, the first shortcut computation may become a bottleneck of a whole smoothing pipeline, and this will consequently decrease the speed of our trajectory smoothing. In this case, we need ways to estimate torque and orientation in parallel to utilize our smoothing method.



(A) (B)

(C) (D)

(E) (F)

FIGURE 3.8: Realtime Motion Planning and Smoothing on a physical robot. The robot loops between point A and point B while the experimenter randomly introduces an obstacle on the robot path. Purple points: raw point cloud from Kinect v2. Blue boxes: occupied voxels obtained by filtering the raw point cloud. Green line: piecewise linear trajectory output by realtime PRM. Pink line: trajectory smoothed in real time by our smoother. (a) Initial trajectory is planned and the robot starts moving. (b) The experimenter introduces the obstacle *after* the robot has started moving. (c, d, e, f) As the obstacle approaches and collides with the planned trajectory, replanning with smoothing is triggered and the robot smoothly avoids the obstacle. The full experiment can be viewed at `https://youtu.be/XQFEmFyUaj8`.

Chapter 4

Time-Optimal Path Tracking with ISO Safety Guarantees

4.1 Introduction

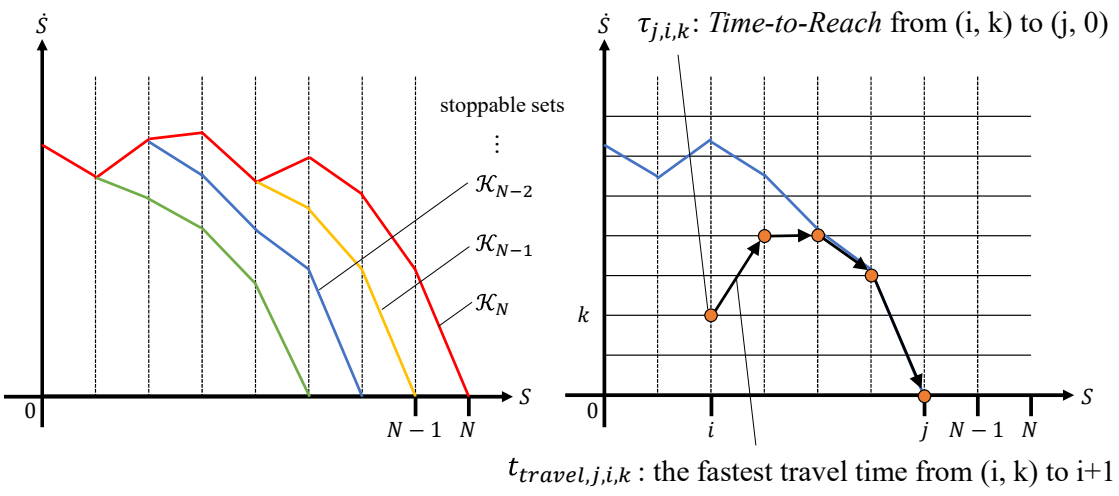
Productivity is crucial in robotic automation, whereas the safety of human workers who work with collaborative robots side-by-side must be ensured. To guarantee the safety of human workers who work with collaborative robots side-by-side, ISO standards define four technical specifications [3, 5]. Among them, in terms of performing automation tasks, they regulate the robot's speed when human workers approach in Category 3 - Speed and separation monitoring (SSM), and the robot's force and torque to minimize injuries caused by collision in Category 4 - Power force limiting (PFL). Our focus in this chapter is to both guarantee the safety in the context of SSM and maximize the speed/productivity simultaneously. Specifically, we consider a time-optimal trajectory/path tracking problem where a robot moves along a given path as *fast* as possible, and the robot *safely* stops at the time instant of the collision with humans not to harm them.

One of the state-of-the-art methods is proposed by Zanchettin et al. [106], which computes the maximum velocity on every control cycle based on current states such as positions and velocities of a robot and obstacles during the path tracking. This method does not require the entire tracking path to be given before the execution. Therefore, advantageously, the path is switchable at runtime to, for example, generate an evasive or reactive motion as long as it satisfies continuity.

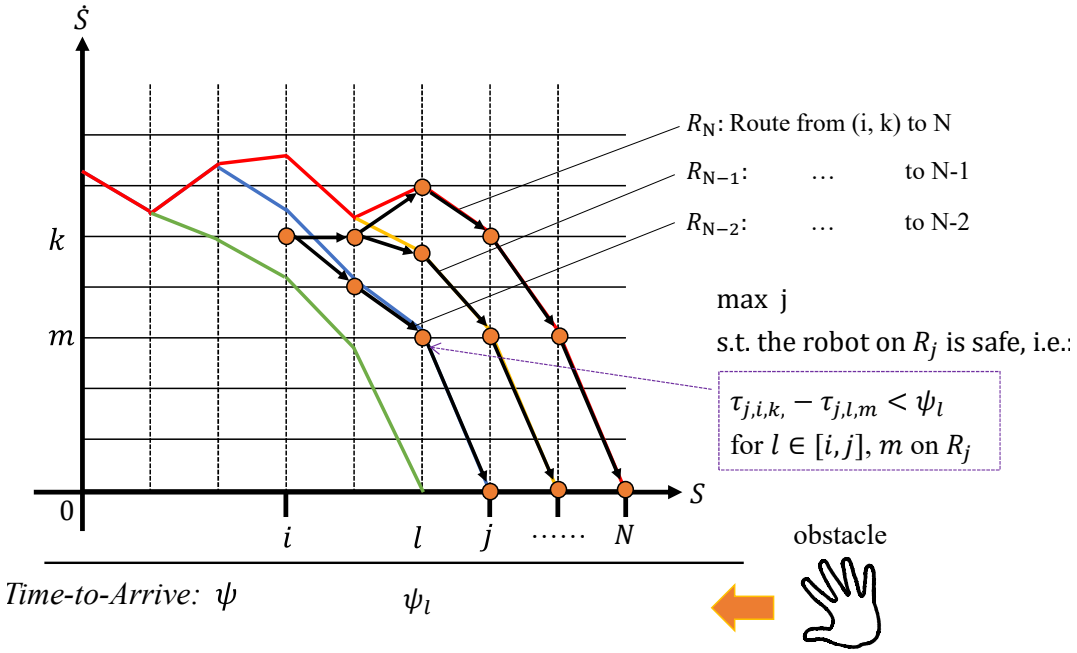
However, it is also a disadvantage because the method does not exploit the entire path information and does not fully leverage the capability of the robot, which makes the system conservative and less productive.

This chapter proposes a time-optimal control method where a robot moves the fastest, and yet safely, satisfying given constraints, founded on time-optimal path parameterization based on reachability analysis (TOPP-RA [15]) and Dynamic Programming. Before executing a trajectory, we scan the path and pre-compute *Time-to-Reach*, the time to go from all the discretized states and stop at all the waypoints along the path. Then, at runtime, using the computed *Time-to-Reach*, we calculate the fastest velocity of the robot on every control cycle guaranteeing that the robot can stop anytime before collision happens (see Fig. 4.1 for illustration). As TOPP-RA handles generalized second-order constraints, our method inherits that characteristic and we can apply any second-order kinodynamic constraints, including joint velocities, accelerations, and torque constraints. In addition, we also propose the method to accelerate the pre-computation of *Time-to-Reach* dramatically by solving one-dimensional linear programming in parallel on GPU, which allows the whole pipeline to be executed at runtime.

This chapter is organized as follows. Our time-optimal path tracking method is presented in Section 4.2, followed by the explanation of the parallelized pre-computation phase on GPU. In Section 4.3, we prove the time-optimality of our method. In Section 4.4, we evaluate our method in three sets of experiments. First, we show that our control policy is less conservative than state-of-the-art safe robot control methods in a 1-D car simulation. Then, the acceleration effect in the pre-computation phase by the parallelization on GPU is examined. Finally, we demonstrate a real-time control of a 6 DoF robot in simulation where the robot moves back and forth between two positions and safely stops before colliding with randomly-moving obstacles. In Section 4.5, some concluding remarks are made with some directions for future work.



(A)



(B)

FIGURE 4.1: Schematic illustrations of the pre-computation phase (a) and the execution phase (b). In the pre-computation phase, the algorithm computes *stoppable sets* ((a)-left) backward from each stage, and compute *Time-to-Reach* as a Dynamic Programming ((a)-right). In the execution phase, based on the current robot states and obstacle positions, the algorithm selects the best *stoppable sets* and route, and calculates the fastest control inputs on every control cycle.

4.2 Time-Optimal Path Tracking with Safety Guarantees

4.2.1 Conceptual Introduction of TOPP-RA

Our method originates from TOPP-RA [15]. Firstly, we provide a conceptual introduction to TOPP-RA to help readers understand. For mathematical and rigorous details of TOPP-RA, we refer readers to the original publication [15].

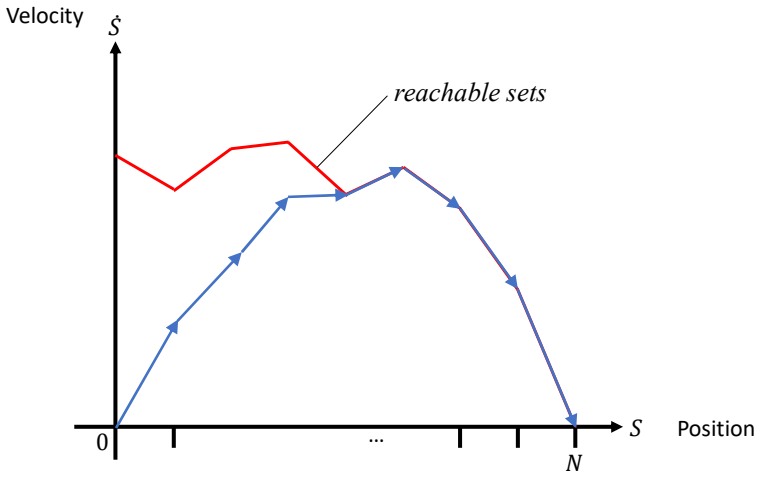


FIGURE 4.2: A conceptual illustration of TOPP-RA.

TOPP-RA addresses the challenge of determining the fastest traversal for a robot to track a given trajectory satisfying constraints. In Fig. 4.2, we represent position along the path for a robot to track on the x-axis, and velocity on the y-axis. The objective is to find the best transition from the start point (located at the origin of the figure) to the goal point (the end point of the path at zero velocity). The higher speed is better, aiming at the time optimality. Please note that, in general, due to kino-dynamic constraints, the fastest and feasible transition is not a simple triangular or trapezoidal form.

To obtain the time-optimal path parameterization, TOPP-RA discretizes the path into $N + 1$ stages, and computes the *reachable sets* backward from the goal point. Within this region, the robot can reach the goal without violating constraints. Subsequently, the algorithm greedily computes the fastest travels from the starting point forward within the region. This approach is proven to be time-optimal.

The pivotal finding in TOPP-RA is, following algebraic transformations, "path-projected dynamics is a discrete-time linear system with linear control-state inequality constraints". Considering a robot configuration q and a geometric path $P = q(s)_{s\in[0,s_{end}]}$ in the configuration space, differentiation yields the following relationships: $\dot{q} = q'\dot{s}$, $\ddot{q} = q''\dot{s}^2 + q'\ddot{s}$. Introducing $x = \dot{s}^2$ (squared velocity) and $u = \ddot{s}$ (input), $\dot{q}$ and $\ddot{q}$ can be linearly formulated in terms of x and u. Consequently, in fact, the second-order constraints can be equivalently represented as linear equations of x and u at every stage s. Moreover, the relationship $\frac{dx}{ds} = 2u$ leads to the discretized linear relation at two consecutive stages, s_i and s_{i+1}: $x_{i+1} = x_i + 2\triangle_i u_i$ (where $\triangle_i = s_{i+1} - s_i$). These linear equations allow us to solve the backward pass and forward pass in TOPP-RA as a set of Linear Programming (LP) problems about x and u, which can be solved efficiently. More specifically, in the backward pass, two-dimensional LPs are solved at every stage s, determining 'in which state x and with what input u the robot can reach the goal'. The forward pass computes the fastest travel by solving one-dimensional LPs about u at every stage s, considering 'what is the fastest/biggest input u that can be applied in the pre-computed reachable set'.

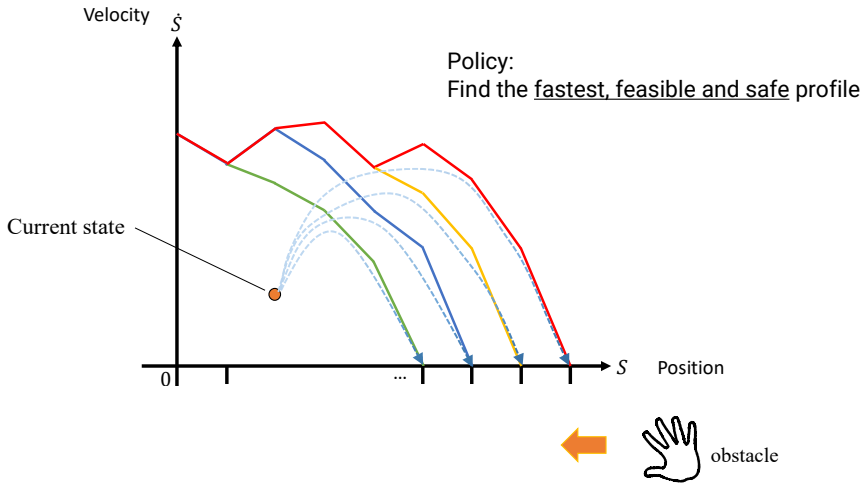


FIGURE 4.3: A conceptual illustration of our proposed safe time-optimal path tracking.

In a dynamic environment, safety becomes imperative as well as speed. Fig. 4.3 illustrates how we achieve both safety and time-optimality extending TOPP-RA formulation. Initially, we precompute the reachable sets for each stage beforehand. Given the set of the reachable sets, we calculate the time-optimal profiles for each of them, and select the fastest and safe one among them at runtime. To assess safety

for each profile, we check whether the robot can traverse all waypoints earlier than any obstacles in the environment. However, this formulation involves numerous one-dimensional LPs in multiple forward pass computations, leading to a computationally intensive algorithm. Our approach solves this problem by computing one-dimensional LPs in parallel on GPU, which enables the whole pipeline to be executed at runtime, nearly in real-time.

4.2.2 Terminologies and definitions in TOPP-RA

Before diving into more technical details, we briefly describe the terminologies and definitions in TOPP-RA that we use in this chapter. Consider an N-DoF robot system, whose configuration is represented as $q \in \mathbb{R}^n$. A geometric path $P = q(s)_{s\in[0,s_{end}]}$ in the configuration space is given where $q(s)$ is a piece-wise C^2-continuous. A *time parameterization* is a piece-wise C^2 increasing scalar function $s(t) : [0, T] \to [0, s_{end}]$.

As in [15], we discretize the interval $[0, s_{end}]$ into N segments and $N + 1$ stages:

$$0 =: s_0, s_1, ..., s_{N-1}, s_N := s_{end} \tag{4.1}$$

Let u_i be a constant path acceleration over the interval $[s_i, s_{i+1}]$ and by x_i the squared velocity $\dot{s}_i^2$ at the i stage, the following relation is derived through algebraic operations:

$$x_{i+1} = x_i + 2\triangle_i u_i, \qquad i = 0...N-1 \tag{4.2}$$

where $\triangle_i = s_{i+1} - s_i$. The generalized constraints in a discretization scheme are considered in a time-parametrization technique:

$$\mathbf{a_i}u_i + \mathbf{b_i}x_i + \mathbf{c_i} \in \mathscr{C}_i \tag{4.3}$$

where coefficients and terms are derived through manipulator's kinematics/dynamics equations and constraints at each stage.

We also borrow the ideas of the i-stage set of *admissible* control-state pairs $\Omega_i := \{(u, x)|\mathbf{a_i}u + \mathbf{b_i}x + \mathbf{c_i} \in \mathscr{C}_i\}$, i-stage *admissible controls* $\mathcal{U}_i(x) := \{u|(u, x) \in \Omega_i\}$. Let $\mathcal{I}$ a set of states, the ideas of i-stage one-step set $\mathcal{Q}_i(\mathcal{I})$ and i-stage *controllable set* $\mathcal{K}_i(\mathcal{I}_N) := \mathcal{Q}_i(\mathcal{K}_{i+1}(\mathcal{I}_N))$ are adopted as well. Intuitively, *admissible* means,

given a state x_i, the control input u_i satisfies the constraints eq. (4.3) and can be applied to the system, whereas *controllable* means there exists control(s) to steer the robot from its state towards the goal state(s) under constraints. See [15] for details.

4.2.3 Problem Formulation

The problem we solve here is to: *"compute the time-optimal control for the robot to track a path under constraints, while the robot must stop when it is not safe"*. Specifically, following the concept of Speed and Separation Monitoring (SSM) in ISO standards [5], "safety" in this chapter is defined as "the robot is static when the minimum distance between the robot and obstacles is under a given protective distance". In other words, a human can touch the robot only when it stops.

We make the following assumptions:

- A geometric path P to track is given.
- The minimum distances between a robot and all dynamic obstacles are monitored, or the field of view of sensors is large enough to detect and safely handle unknown obstacles.
- Maximum velocities of dynamic obstacles are given.

Note that the modeling of obstacles is not a requirement. Raw sensor data, such as point cloud, can be used as long as the real-time computation of the minimum distance between a robot model and captured points on dynamic obstacles is possible. Additionally, *if* you have an estimation of obstacles' motions, you can incorporate it in eq. (4.9), which will produce a less-conservative motion. However, in this chapter, we do not trust and rely on such predictions as discussed in Section 2.3.3.

Our technique can be separated into a pre-computation phase and an execution phase. In the pre-computation phase, we compute *controllable sets* for a robot to stop at each stage. We call these sets as *stoppable sets*. In the execution phase, we select the best *stoppable sets* among the set of *stoppable sets* to go as far as possible, and compute the time-optimal control input. Note that the pre-computation phase takes only about 0.5 s practically (see Section 4.4.3), such that the whole pipeline

can be executed in near real-time. We explain these two phases step-by-step in the following sections.

4.2.3.1 Pre-computation Phase

Fig. 4.1a illustrates the pre-computation phase. Firstly, we compute stoppable sets $\mathcal{K}_j$ for each stage $j \in [1, N]$ (the left side of Fig. 4.1a). This step is almost equivalent to the "backward pass" in TOPP-RA [15], but the difference is that we process the backward pass from each stage. The stoppable sets are drawn in different colors in Fig. 4.1a.

Secondly, given the number of discretizations in velocity space M, we determine the discretization size of $\dot{s}$ with the following equation, used later for dynamic programming:

$$\delta v := \frac{\sqrt{\max \mathcal{K}}}{M} \tag{4.4}$$

where $\mathcal{K}$ is the union of all the stoppable sets $\mathcal{K}_j$ (which is actually $\mathcal{K}_N$, see eq. (4.15)). Please note that we discretize the velocity space uniformly, not the x i.e. the squared $\dot{s}$.

Then, for each stoppable set, we compute the fastest travel time $t_{travel,j,i,k}$, which is the fastest time to control the robot at stage i with the velocity $k\delta v$ to the next $(i+1)$ stage under constraints, for $i \in [0, j-1], k \in [0, M]$ (see the right side of Fig. 4.1a). This step is equivalent to the "forward pass" in TOPP-RA, but we compute the greedy controls for all discretized states s and $\dot{s}$ for each stoppable set.

$$\begin{aligned} u^* &:= \max u, \text{ s.t.:} \\ x_k + 2\triangle_i u \in \mathcal{K}_{j,i},\ u \in \Omega_i, x_k &= k\delta v \end{aligned} \tag{4.5}$$

$$x^*_{k,i+1} := x_k + 2\triangle_i u^* \tag{4.6}$$

$$t_{travel,j,i,k} = \frac{2\triangle_i}{\sqrt{x^*_{k,i+1}} + \sqrt{x_k}} \tag{4.7}$$

The complexity of this computation is $O(N^2 M)$ and the bottleneck is 1-D Linear Programming of eq. (4.5). We propose a parallelization technique for accelerating this step in Section 4.2.4.

Finally, we apply dynamic programming to compute *Time-to-Reach* $\tau_{j,i,k}$, which is the fastest time to go from the starting stage i at the velocity $k\delta v$ and stop at the stage j. We also record the fastest travel $\rho_{j,i,k}$ for each grid point. This processing enables parallelization at the runtime execution phase.

$$\begin{aligned} \rho_{j,i,k} = \left\lfloor \left(\frac{\sqrt{x^*_{k,i+1}}}{\delta v}\right) \right\rfloor \tau_{j,j,all} = inf, \quad \tau_{j,j,0} = 0 \\ \tau_{j,i,k} = \tau_{j,i+1,\rho_{j,i,k}} + t_{travel,j,i,k} \end{aligned} \tag{4.8}$$

The overall algorithm is described in algorithm 1. Note that floor $\lfloor\rfloor$ and ceil $\lceil\rceil$ operations are used in calculating velocity indices in a "safe" manner, i.e., not violating constraints and not over-estimating the fastest travel time.

Now we have *Time-to-Reach* τ and the fastest travel ρ, used in the execution phase for real-time control explained in the next section.

4.2.3.2 Execution Phase

Fig. 4.1b illustrates how to compute the time-optimal controls. Let $d_{protective}$ denote a given protective distance. Consider that the robot is at the position of the stage i with the velocity v_i.

First, we compute the minimum distances $d_{l,q}$ between a robot that locates at every stage l and each obstacle q, and then calculate the *Time-to-Arrive* ψ_l for each l stage:

$$\psi_l = \min_q \frac{d_{l,q} - d_{protective}}{v_{q,max}} \tag{4.9}$$

where $v_{q,max}$ denotes the maximum velocity of obstacle q in Cartesian space.

To control a robot to go as fast as possible, we select the farthest stage to stop, where the robot stops and then (possibly) collides:

$$\begin{aligned} j_{stop} = \max_{j \in [i,N]} j \\ \text{s.t.: } (\tau_{j,i,k} - \tau_{j,l,m}) < \psi_j \ \forall l \in [i,j], \text{m on the route } R_j \end{aligned} \tag{4.10}$$

where the stopping route R_j can be obtained by tracing the fastest travel ρ_j.

Finally, we apply the exact "forward pass" of TOPP-RA from i to j_{stop} to compute the time-optimal path parameterization using the stoppable set $\mathcal{K}_{j_{stop}}$, and apply the fastest velocity s_i at the stage i to the robot. We repeat this execution process for every control cycle. Practically, this runs within 6 milliseconds for 500 stages in a 6 DoF robot experiment in Section 4.4.3, which is suitable for real-time control.

The farthest stage calculation has a computational cost of $O(N^2)$, but it is computationally fast enough, typically completing in 1–2 ms at most. In cases where longer trajectories exist and/or fine discretization is required, we can parallelize this process owing to the formulation with *Time-to-Reach*. A parallelized version enumerates all the routes to be traced beforehand, and evaluates eq. (4.10) for every iteration in a batched and scalable manner. We observed that the parallelized algorithm runs in less than 1 ms at worst on GPU, with additional costs in the pre-computation phase primarily for memory allocation and data transfer.

In Section 4.3, we show that our method is time-optimal in the sense that *For any robot motion that is strictly faster than the motion recommended by our policy, there exists a human motion that results in a collision with the robot in a non-stationary state.* See 4.3 for the proof.

4.2.4 Parallel 1-D Linear Programming on GPU

Our primary contribution is that we accelerate the pre-computation phase computation by 10x, which allows the whole pipeline to be executed at runtime. The key idea is the parallelization of 1 Dimensional Linear Programming (LP) (eq. (4.5)) on GPU. The 1-D LP can be transformed into the following form:

$$\begin{aligned} &\max\ u \\ &s.t.\ a_i u \leq b_i, a_i \neq 0 \end{aligned} \tag{4.11}$$

Note that the notations a_i, b_i, and c are re-used in different meanings from eq. (4.3). This can be easily solved as follows [174]. Constraints are classified into two groups - two half-spaces C^+, C^-:

$$\begin{aligned} C^+ &= \{i | u \leq b_i/a_i\} \\ C^- &= \{i | u \geq -b_i/a_i\} \end{aligned} \tag{4.12}$$

Algorithm 1 Compute Time-to-Reach as a Dynamic Programming

Given M: the number of discretization in x
Output $\mathcal{K}$: Stoppable sets - a set of Controllable sets $\mathcal{K}_j$ for stopping at stage j, where $j \in [0, N+1]$,
Output $\tau_{j,i,k}$: Time-to-Reach starting from the stage i and the velocity index k, and stopping at stage j, where $j \in [1, N], i \in [0, N-1], k \in [0, M]$

/* Backward pass for stopping sets */
for $j \in [N, 1]$ **do**
  $\mathcal{K}_{j,N} := 0$
  for $i \in [N-1, 0]$ **do**
    $\mathcal{K}_{j,i} := \mathcal{Q}_i(\mathcal{K}_{j,i+1})$
  end for
end for
$\delta v := \frac{\sqrt{\max \mathcal{K}}}{M}$
/* For each stoppable set */
for $j \in [N, 1]$ **do**
  /* Dynamic Programming */
  $\tau_{j,j,all} := \inf$
  $\tau_{j,j,0} := 0$
  for i in $[j-1, 0]$ **do**
    $x_i^-, x_i^+ \leftarrow \mathcal{K}_{j,i}$
    $l := \left\lceil \frac{\sqrt{x_i^-}}{\delta v} \right\rceil$, $m := \left\lfloor \frac{\sqrt{x_i^+}}{\delta v} \right\rfloor$
    for k in $[l, m]$ **do**
      /* Compute the fastest travel time at stage i */
      $u^* := \max u$, s.t.: $x_k + 2\triangle_i u \in \mathcal{K}_{j,i}$, $x_k = (k\delta v)^2$, $u \in \Omega_i$
      $x^*_{k,i+1} := x_k + 2\triangle_i u^*$
      $v := \sqrt{x_k}$, $v^*_{k,i+1} := \sqrt{x^*_{k,i+1}}$
      $t = 2\triangle_i / (v^*_{k,i+1} + v)$
      /* Compute Fastest Travel and Time-to-Reach */
      $\rho_{j,i,k} = \left\lfloor (\frac{v^*_{k,i+1}}{\delta v}) \right\rfloor$
      $\tau_{j,i,k} := \tau_{j,i+1,\rho_{j,i,k}} + t$
    end for
  end for
end for

Then, when $\alpha \leq \beta$ where $\alpha := \max\{-b_i/a_i | i \in C^-\}$ and $\beta := \min\{b_i/a_i | i \in C^+\}$, the 1-D LP is feasible and its optimal solution u can be found as:

$$u^* = \beta \tag{4.13}$$

Here, we only use basic arithmetic operations and $\max, \min$ whose performance for parallel computation is pretty optimized in GPGPU, and computationally efficient APIs are provided by GPGPU frameworks such as PyTorch [175]. No if/else switching is required that causes warp divergence leading to poor performance [176].

4.3 Proof of Optimality

In this section, we prove that our method is time-optimal in the sense that *For any robot motion that is strictly faster than the motion recommended by our policy, there exists a human motion that results in a collision with the robot in a non-stationary state.*

First, we prove the following theorem:

Theorem 4.1. *The stoppable sets $\mathcal{K}_{j,i}$ always satisfies the following:*

$$\mathcal{K}_{j,i} \subseteq \mathcal{K}_{k,i} \ s.t. \ \forall j < k, \forall i \leq k \tag{4.14}$$

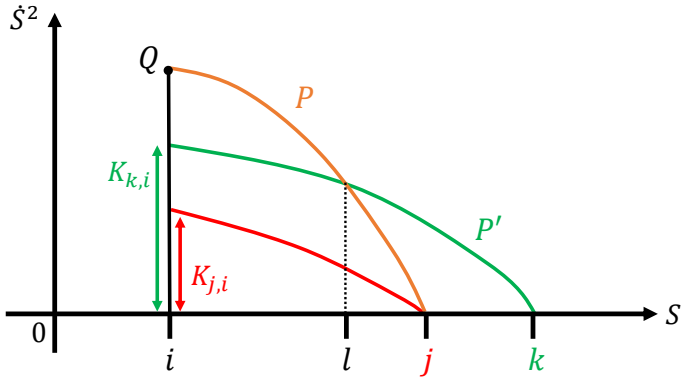


FIGURE 4.4: A figure for the proof of inclusion relationship between the stoppable sets.

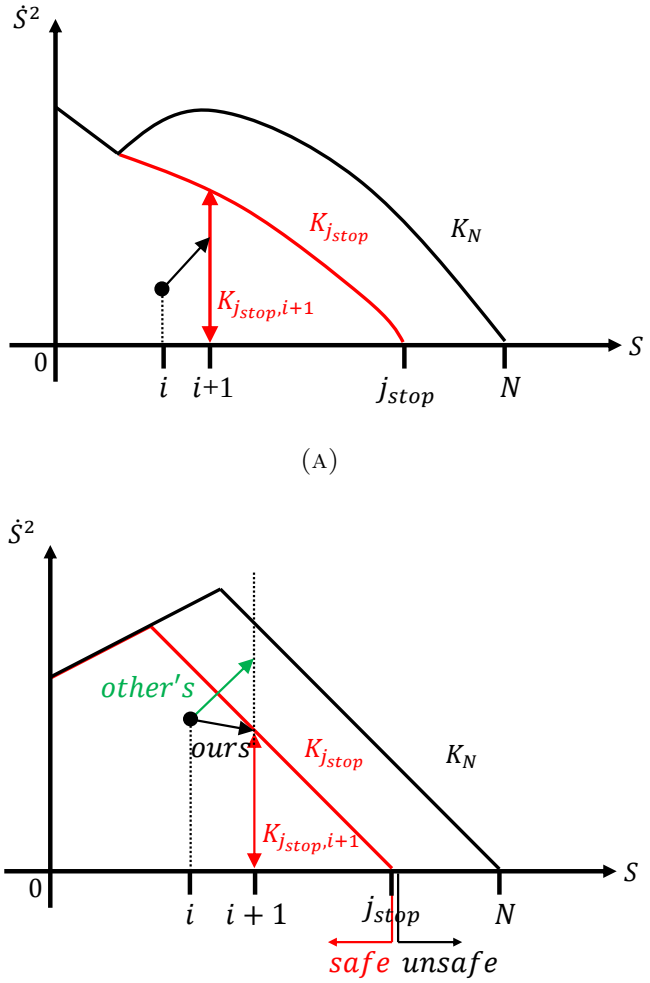


FIGURE 4.5: Figures for the proof of the optimality of our method.

Proof. Assume by contradiction that there exists a velocity $\dot{s}^2$ that is $\mathcal{K}_{j,i}$ but not in $\mathcal{K}_{k,i}$ (the point Q in Fig. 4.4). By definition of $\mathcal{K}_{j,i}$, there exists a profile P (the orange line) starting from that $\dot{s}^2$ (Q) and that stops at j. By continuity of the boundaries (see Appendix A of TOPP-RA paper [15]), there exists a time l where the profile intersects the boundary of $\mathcal{K}_k$ (the green profile). We construct a profile P' by gluing together the beginning of the orange profile (P) and the end of the green profile. We have then proved that $\dot{s}^2 \in \mathcal{K}_{k,i}$, which contradicts the initial assumption. □

From Theorem 1, the following can be derived:

$$\mathcal{K}_{j,i} \subseteq \mathcal{K}_{k,i} \ldots \subseteq \mathcal{K}_{N,i}, \forall i \tag{4.15}$$

Based on the above, we derive the following theorem:

Theorem 4.2. *For any robot motion that is strictly faster than the motion recommended by our policy, there exists a human motion that results in a collision with the robot in a non-stationary state.*

Proof. Suppose the robot is at i, our method selects the stoppable set $\mathcal{K}_{j_{stop}}$ and computes a control at stage i using TOPP-RA's one-step greedy forward pass. Assume by contradiction that there exists a strictly faster and safe control u_i^{**} by other motion policy.

Since the robot must be controlled to stop at the final destination (s_N), any motion policies keep the robot state s^2 satisfies $\dot{s}_l^2 \in \mathcal{K}_N$ s.t. $(i < l \leq j_{stop} \leq N)$. In addition, any control input driven by all the other policies must also be *admissible*, i.e., $u \in \mathcal{U}_i(x)$, otherwise the policy violates the given constraints.

From eq. (4.10) (our stoppable set selection strategy) and eq. (4.15), all the possible states $\dot{s}_l^2$ s.t. $i < l \leq N$ can be categorized into the following only two sets:

- $\mathcal{K}_{j_{stop},l}$, that are 'safe' (guaranteed as no collision), and
- $\mathcal{K}_{N,l} \cap \overline{\mathcal{K}}_{j_{stop},l} \triangleq \mathcal{K}_{unsafe,l}$, that are not 'safe' (there exists a human motion that can collide with the robot).

At the forward step of the execution phase at the stage i, there are two cases where:

1. $\mathcal{K}_{j_{stop},i+1}$ becomes a non-active constraint.
2. $\mathcal{K}_{j_{stop},i+1}$ becomes an active constraint.

In the case 1: TOPP-RA's forward pass generates the most greedy control input $u_i^* \in \mathcal{U}_i(x)$ via maximization of u, which contradicts the assumption that there exists the strictly faster control u_i^{**} than u_i^* (see Fig. 4.5a for illustration).

In the case 2: u_i^{**} drives the robot state x_{i+1} at the stage $(i+1)$ out of $\mathcal{K}_{j_{stop},i+1}$ with $x_{i+1} = x_i + 2\triangle_i u_i^{**}$, which is in $\mathcal{K}_{unsafe,i+1}$. This is unsafe and contradicts the initial assumption that u_i^{**} is safe (see Fig. 4.5b for illustration). □

4.4 Experiments and Results

4.4.1 Comparison with Existing Method

Firstly, we compare our method with the existing state-of-the-art method (Zanchettin's) [106] in a simple 1-D car simulation (Fig. 4.6). We put two cars at the same position (0 m) and they start to move from left to right at the same time towards the same goal position (25 m) controlled by each method. The car above represents Zanchettin's method and the below one is ours. Cars have the same velocity and acceleration limits (20 m/s and 100 m/s^2 respectively). For our method, the protective distance is set to 0. A green wall moves from right to left to hinder the cars at its constant speed (20 m/s), and when it collides with the faster car, waits for 1 sec, and moves back to the right direction. The stopping time in Zanchettin's method is manually optimized to (0.55 s) to be small enough not to have an infeasible optimization problem.

We show the result in Fig. 4.7. At first, two cars accelerate at their maximum acceleration to their maximum speed. Then, while the Zanchettin's car gradually decelerates and stops as the wall approaches way before the wall collides, ours stops almost exactly when the distance becomes 0 at its maximum deceleration. As a result, our car arrives earlier than Zanchettin's. The experiment can be viewed at `https://youtu.be/SHwyOOU3X2A`.

We summarize the following observations about Zanchettin's method from this experiment: 'Stopping time' needs to be large and conservative enough to consider the case when a robot and an obstacle come at their max speed. Otherwise, the optimization problem is not always feasible e.g., the robot cannot decelerate enough to stop in time. This property conservatively constrains the velocity from the upper side with the constraint (9b) of the LP problem (9) in [106]. Moreover, the authors claim that '$\dot{q}_{k+1} = 0, \delta_k = 0$ is always a solution', but it is not true because, due to lower acceleration limits, the optimization problem can be infeasible when the robot moves too fast to decelerate enough within a stopping time to avoid unsafe collision.

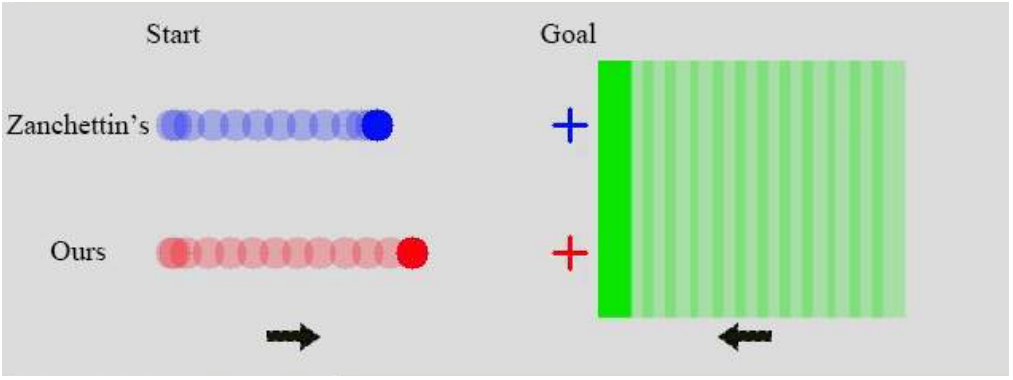


FIGURE 4.6: 1-D car simulation setting for comparison

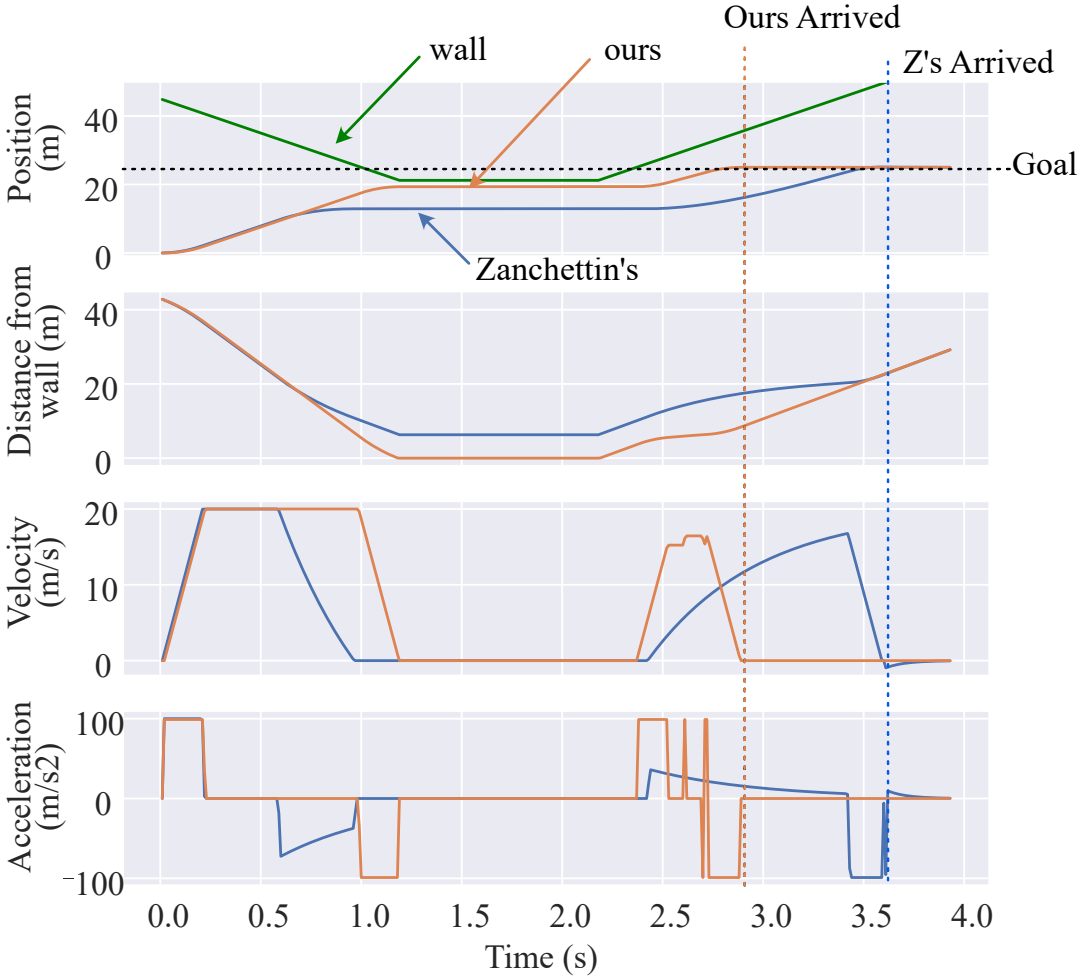


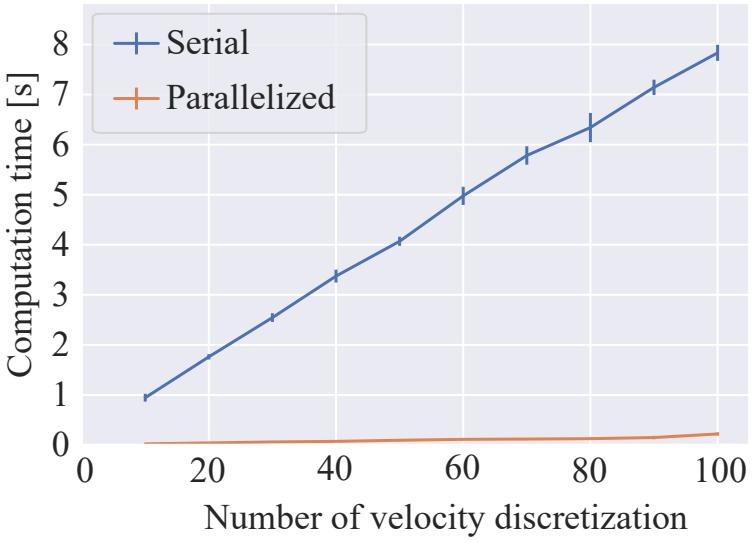
FIGURE 4.7: Comparison of Zanchettin's and Ours - Positions, Distance between cars and a wall, Velocities and Accelerations in a time series. See the video of this simulation at `https://youtu.be/SHwyOOU3X2A`.

4.4.2 Parallel 1-D Linear Programming on GPU

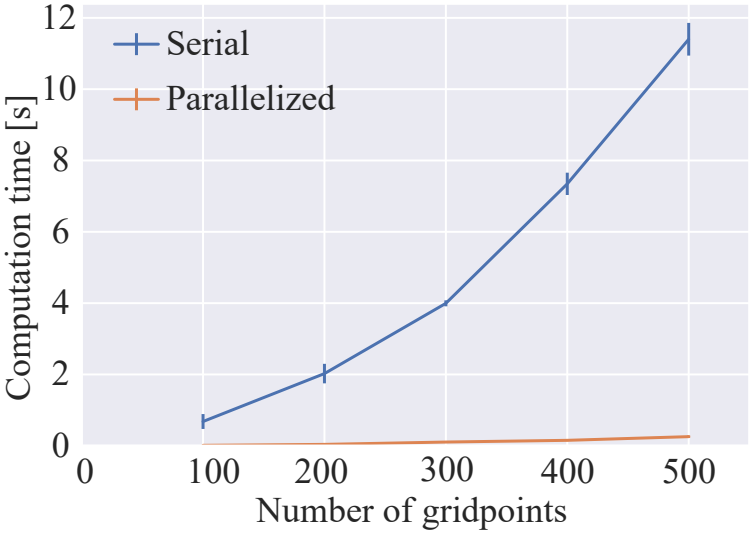
Secondly, we experimentally show the performance of parallel 1-D Linear Programming solver running on GPU, compared to serial computation with vanilla TOPP-RA on CPU. The performance is measured with the computation time from the line 13 to the end of algorithm 1 that is right after the computation of stoppable sets $\mathcal{K}$ and includes Dynamic Programming, changing the number of stages N and the number of velocity discretization M. We randomly generate three 6-DoF waypoints, interpolate them with a spline, and compute its *Time-to-Reach* under randomly generated joint velocity limits and acceleration limits for 10 trials. Our method is implemented in Python with PyTorch, and the serial processing version is in Python with Cython. The serial version is optimized to use memorization of the solutions of 1-D LP problems for the case when $\mathcal{K}_{j_1,i}$ and $\mathcal{K}_{j_2,i}$ $(j_1 \neq j_2)$ are the same. This experiment is executed on a single machine, whose CPU is Intel® Xeon® W-2145 and GPU is GeForce GTX 1080 Ti. We fix $N = 300$ when we change M, and $M = 50$ when we change N. Fig. 4.8 clearly shows that the computation time increases linearly to N and to the square of M. The time for pre-computation with serial processing takes 1 sec even with the smallest numbers of discretization ($N = 300, M = 10$ and $N = 100, M = 30$) in this experiment, which is too coarse and useless for real applications. On the other hand, our parallelized 1-D LP solver dramatically outperforms serial processing and takes only 0.25 sec even with the largest number of discretizations in the experiment ($N = 500, M = 30$). This result clarifies our contribution of allowing the whole pipeline to be adopted at runtime, nearly in real-time.

4.4.3 Simulation on a 6-Dof Industrial Robot

Finally, we conducted a 6 DoF robot experiment in simulation (Fig. 4.9). In Fig. 4.9a, A robot moves back and forth between two positions while a dynamic obstacle moves in a bouncy, jig-zag manner in front of the robot at the speed of 1.6 m/s, the conservative human's speed as reported in [5]. The protective distance is set to 0 for demonstration. For fast minimum-distance computation, we simplify the robot by modeling all the robot links (not only the end-effector) with their enclosing spheres and compute the distances between the centers of the spheres and the center of the obstacle. Our algorithm is implemented on top of



(A) Effect of the number of velocity discretization M



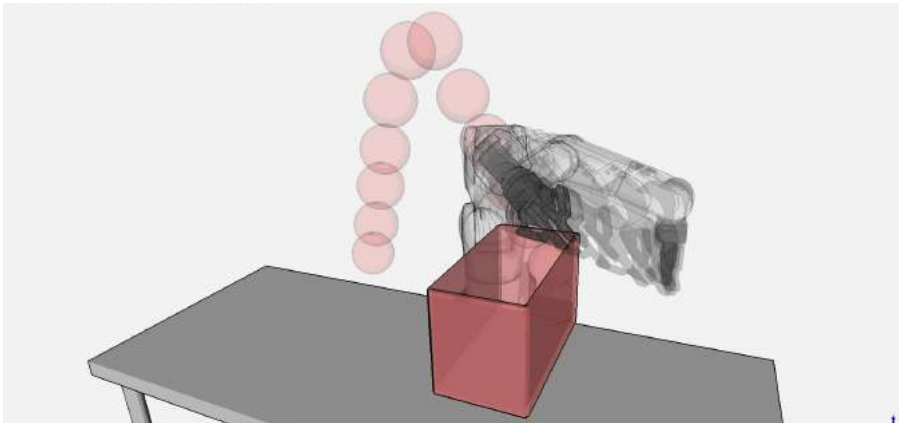
(B) Effect of the number of stages N

FIGURE 4.8: *Time-to-Reach* computation time comparison, Serial processing v.s. our Parallelized 1-D Linear Programming solver.

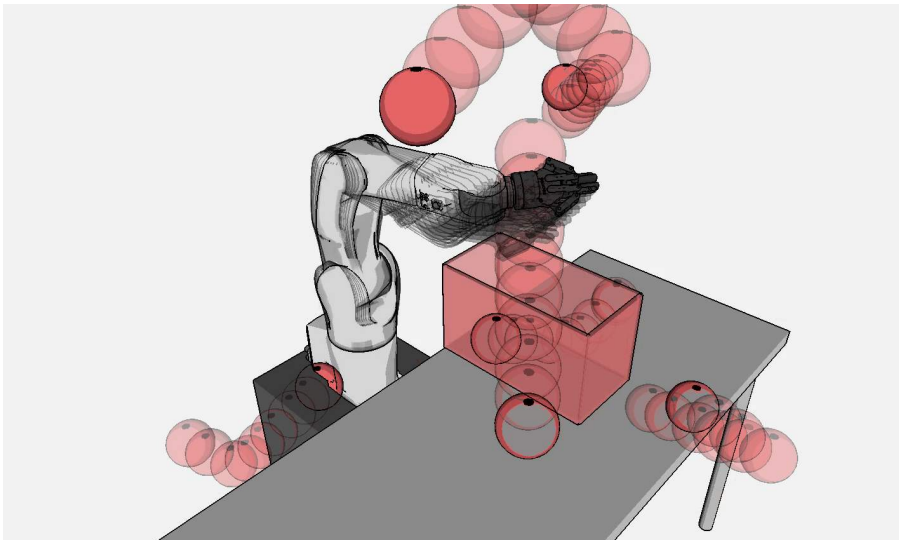
OpenRAVE[173] and the path the robot tracks is computed by an off-the-shelf Bi-RRT planner and a parabolic smoother provided by OpenRAVE. The program runs on a laptop with AMD Ryzen 9 4900HS and NVIDIA GeForce RTX 2060 with Max-Q Design. The number of velocity discretization M is set to 30 and the minimum number of stages N is set to 500, resulting in 517 stages proposed by TOPP-RA library [177]. Throughout the experiment, the robot moves, stops safely at its maximum accelerations (± 20 rad/s^2) when the obstacle approaches ([A]–[B] in Fig. 4.10) and collides ([C]–[D] in Fig. 4.10) with the robot, and restarts its motion without violating joint velocity limits and acceleration limits. Our method works even in a cluttered environment where 6 dynamic obstacles move at 1.6 m/s in random directions (Fig. 4.9b). The mean value of total pre-computation for 20 trials is accelerated from 4.03 ± 1.01 s to 0.40 ± 0.09 s by our proposed parallelized 1-D LP solver, which is 10 times faster. Both results include the computation time of stoppable sets (0.20 ± 0.04 sec) that runs on 16 CPU processes in parallel. For reference, when enabling GPU at the execution phase for the farthest stopping stage computation, the pre-computation phase takes 0.54 ± 0.16 s due to additional cost for memory allocation and data transfer. The total computation of the execution phase runs within 6 ms on CPU, and 5 ms on GPU. The experiment can be viewed at `https://youtu.be/ta3lx80jJjk`.

4.5 Summary

In human-robot collaboration, speed and safety are generally in a trade-off relationship, which has led collaborative robots to be conservative and less productive. This chapter proposes a time-optimal control method for a robot to track a path guaranteeing the safety of human workers satisfying given constraints, based on TOPP-RA and Dynamic Programming, according to the SSM framework in ISO standards. Our controller provides time-optimal control inputs at runtime and is strictly less conservative than the state-of-the-art controller guaranteeing collision safety, which is experimentally shown in simulation and is demonstrated in a 6-DoF robot experiment. In addition, the pre-computation phase is accelerated by leveraging GPU and has been reduced down to 0.5 s in the 6 DoF robot experiment, which is 10x faster than that of vanilla implementation in TOPP-RA and can be executed at runtime nearly in real-time.



(A)



(B)

FIGURE 4.9: (a) 6-DoF Robot control experiment in a simulation where the robot moves "safely" in an environment in which 1 dynamic obstacle is randomly moving at its maximum speed. (b) Our method can navigate the robot through a crowded environment with 6 randomly-moving dynamic obstacles. See the video of this experiment at `https://youtu.be/ta3lx80jJjk`

Future work can be in the following directions: The computation of stoppable sets is a sequential process whose computational complexity is $O(N^2)$. It does not fit parallel operations on GPU and now becomes a bottleneck of the entire pre-computation phase. Another problem is space complexity. Current implementation consumes about 5 GB memory on GPU in a robot experiment (Section 4.4.3), and in case we apply more constraints it will increase in $O(a^3)$ where a is the number of coefficients of constraints. Yet another remark is about the smoothness. For example, we observe that, when the human is on the same trajectory of the robot

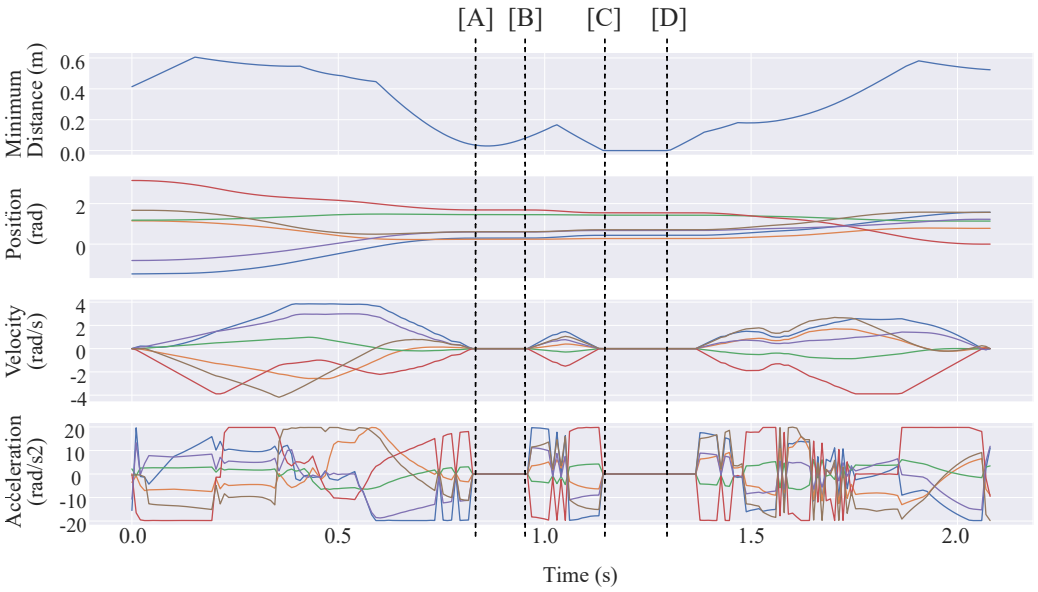


FIGURE 4.10: A result of one trajectory execution, logging Minimum distance from a dynamic obstacle, Joint Positions, Velocities and Accelerations. The robot decelerates at its maximum accelerations (± 20 rad/s^2) and stops (velocity is 0) when the obstacle approaches ([A]–[B]) and collides ([C]–[D]) with the robot, and restarts its motion without violating joint velocity limits and acceleration limits.

and followed by the robot, the robot repeatedly accelerates and decelerates to come to the next waypoint as fast as possible once the human passes a waypoint, resulting in a "bang-bang" acceleration profile (we can see the spikes in Fig. 4.7). However, it would be desirable for the robot to smoothly track its trajectory and follow the human, even if it sacrifices "time optimality". Exploring this *human-aware* path tracking is an interesting future direction.

Chapter 5

Real-time Batched Distance Computation for Time-Optimal Safe Path Tracking

5.1 Introduction

Collaborating with robots while ensuring human safety has been a critical challenge, as slowing down the robot operation to mitigate injuries will impede productivity. To maximize the productivity of collaborative robots while guaranteeing the safety, we have proposed time-optimal path tracking algorithm [178] which runs in real-time and provides the safe and fastest control input with respect to *Speed and Separation Monitoring* in ISO standards [5]. In this path-tracking method, distances between the obstacles and a robot for waypoints along an executing trajectory must be given. Given the distances, the algorithm computes the fastest velocity profile and navigates the robot in a time-optimal manner (Fig. 5.1). Finally, the control input is sent to a robot to follow the derived velocity profile. This whole process must run in every control cycle, which is about 10 ms according to the communication protocol of industrial robots[1].

[1]For example, the control period is 8 ms in the case of DENSO b-CAP communication protocol `https://www.denso-wave.com/en/robot/product/function/b-CAP.html`.

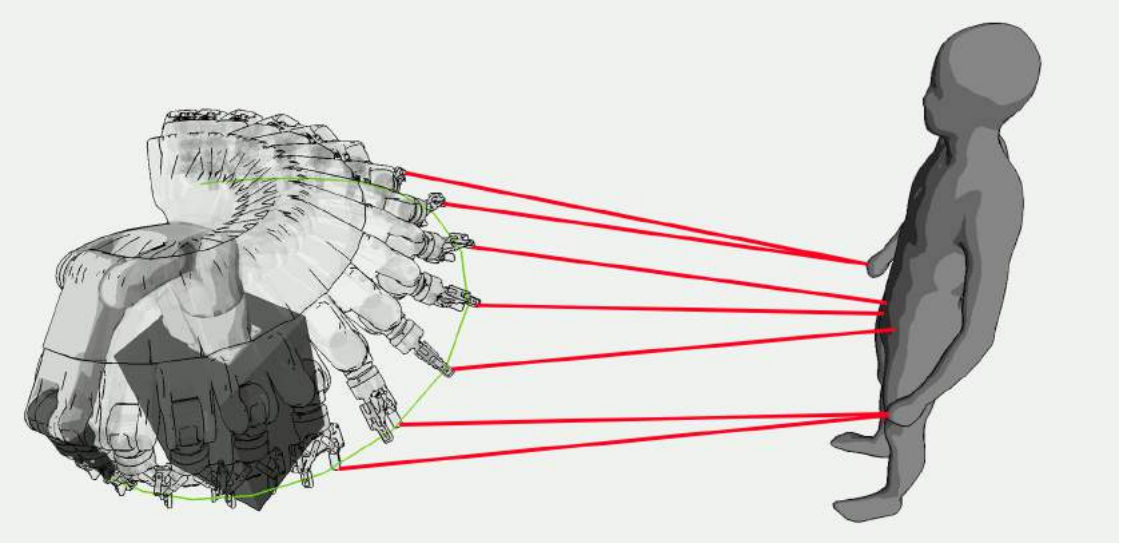


FIGURE 5.1: Problem Setting Overview: Computing distances in real-time, across multiple robot configurations, with precision. See Section 5.1 for more information.

To achieve true optimality in path tracking, precise distances need to be given. In our previous paper, the robot is simplified with spheres, and the distances between the spheres and voxels are computed with *hypot* function using their center positions. Such distance checking with a simplified model can run almost in real time. However, the computed distances are smaller than their actual values due to the simplification, which makes the robot's behavior conservative and exacerbates the productivity of the robot. In contrast, an exact mesh-to-mesh distance checker cannot run in real-time (experimentally, 130 µs per one configuration, 65 ms per one trajectory with FCL [16]). To the best of our knowledge, no existing distance checker is applicable to real-time safety control.

In this chapter, we propose a batched, fast, and precise distance checker based on pre-computed link-local Signed Distance Fields(SDFs) to address this issue. Leveraging GPU parallelization for pre-processing of robot's SDFs, the proposed method is able to check distances for multiple robot configurations within less than 1 ms at runtime. Additionally, a neural approximation of the pre-processing has been proposed, resulting in 2x faster pre-processing. Finally, we experimentally demonstrate that our distance checker actually navigates a robot faster than the method using a robot modeled with spheres in a dynamic, collaborative environment.

This chapter is organized as follows. Section 5.2 presents our parallel distance

computation method and some techniques to reduce the pre-processing time in a constant order including the neural approximation. In Section 5.3, we evaluate the performance of the neural approximation and also examine that the approximation does not affect the precision of distance computation. Then, the experimental comparison is shown for the real-time safe path tracking in a collaborative environment. Finally, we discuss the limitations of our approach and conclude with some directions for future work in Section 5.4.

5.2 Batched Robot SDFs Computation

5.2.1 Overview

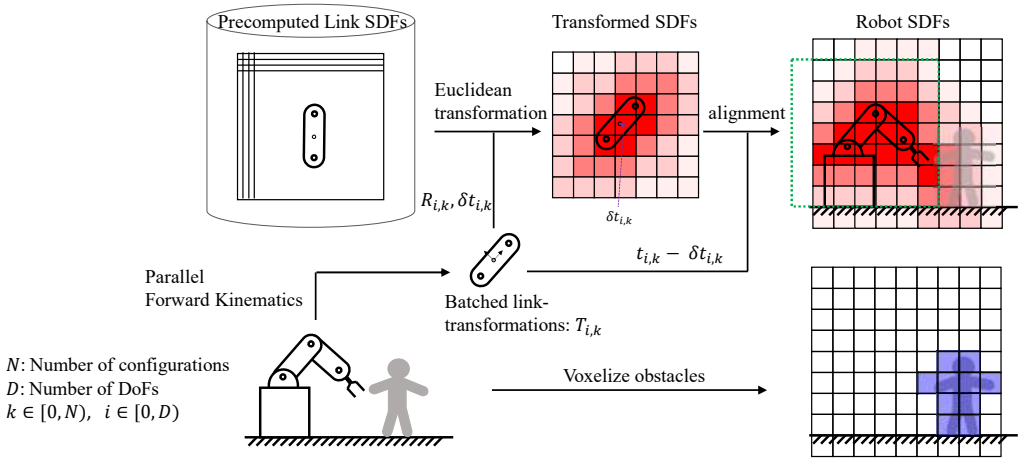


FIGURE 5.2: A pipeline of parallel batched distance checking with pre-computed link-wise signed distance fields (SDFs). This is in 2D for clear illustration, but actual computation is in 3D and in a batched manner. See Section 5.2 for detail.

The pipeline of our parallel distance checking is illustrated in Fig. 5.2. We consider a D-DoF robot and examine distances at C robot configurations ($\theta_{\mathbf{c}} \in \Theta$). The environment is discretized into voxels whose extent is $\mathbf{e_e} = (e_{ex}, e_{ey}, e_{ez})$ and resolution is $\mathbf{r_e} = (r_{ex}, r_{ey}, r_{ez})$. The total number of voxels of the environment V_e is $\prod \frac{2\mathbf{e_e}}{\mathbf{r_e}}$.

At the preprocessing stage, given a robot model, we pre-compute Signed Distance Fields(SDFs) for each link on its local coordinates. We call it as *Link SDFs*. We

refer $\mathbf{e_r}$ to the extent of Link SDFs (e_{rx}, e_{ry}, e_{rz}) and $\mathbf{r_r}$ to the resolution of Link SDFs (r_{rx}, r_{ry}, r_{rz}). The size of the precomputed Link SDFs, $2\mathbf{e_r}$, must be divided by the resolution of the voxelized environment $\mathbf{r_e}$ without residue for alignment operation that is later introduced. The resolution of Link SDFs is arbitrary and recommended to be finely voxelized.

Next, given the configurations c, we compute transformations $T_{i,c}$ of each link i by applying parallel forward kinematics. Then, according to $T_{i,c}$, Link SDFs are transformed and aligned into the voxels of the environment. We call the first euclidean transformation operation "euclidean transformation" and the second alignment operation "alignment". To compute Robot SDFs for each configuration, a minimum value of the transformed Link SDFs for each link and for each voxel is taken. Besides, obstacles in the environment are voxelized. By extracting the distances at the voxels occupied by the obstacles and taking the minimum value for each link, the distance between the robot and the obstacles can be computed.

More specifically, at the "euclidean transformation" stage, we shift the rotated Link SDFs within the half range of the voxel by $\delta t_{i,c} \in (-\frac{r_e}{2}, \frac{r_e}{2})$. And then, at the "alignment" stage, we translate the transformed SDFs by $t_{i,k} - \delta t_{i,k}$ and snap them into the environment voxels. The shift operation is necessary for exact alignment since the position of each link in the environment is not usually at the exact center of the voxel. The transformation can be computed in the scheme of affine grid transformations [179] [2]. $\delta t_{i,c}$ can be computed by following the simple equations:

$$T_{Oi,c} = T_{i,c} - (-e_e) \tag{5.1}$$

$$k_{i,c} = \lfloor T_{Oi,c}/r_e \rfloor - \lfloor e_e/e_r \rfloor \tag{5.2}$$

$$\delta t_{i,c} = T_{Oi,c} - (k_{i,c} \cdot r_e + e_r) \tag{5.3}$$

where $\mathbf{e_e}$ is the 3D extent of environment, $\mathbf{r_e}$ is the 3D resolution of environment, and $\mathbf{e_r}$ is the 3D extent of Link SDFs. The total number of voxels in transformed SDFs V_r is $\prod \frac{\mathbf{e_r}}{\mathbf{r_e}}$.

At runtime, to compute distances against obstacles in the environment based on the Robot SDFs, we voxelize the obstacles and extract the values from Robot SDFs that are occupied by the voxelized obstacles. By reducing the extracted values with

[2] Please refer to pytorch's documentation as well: `https://pytorch.org/docs/stable/generated/torch.nn.functional.affine_grid.html`

min for each configuration c, we can obtain the minimum distance between a robot and the obstacles for each c. This process is fast because it only reads the data on the GPU memory and does not require any calculation.

As an extra bonus, self-collision detection can be done by aligning "in a predefined and alternating the order of checking, paying attention to the robotś kinematics" as in [123], though it is not applied in our experiment since self-collision is usually examined in the motion-planning phase rather than the execution phase.

There can be a variation for reduction of Robot SDFs. For example, if you only need a binary occupancy information of the robot, you can use store boolean values for each voxel, checking whether the distance is greater or less than 0. This will lower the memory consumption by 8 times (= sizeof(float) / sizeof(bool)).

5.2.2 Techniques for computation time reduction in a constant order

We introduce the following techniques to optimize the computation time in a constant order.

5.2.2.1 Euclidean Grid Approximation with A Tiny Neural Network

The computation of euclidean grid transformation mapping is mathematically a matrix multiplication. Given the center positions of grids $p_{j,xyz}$for $j \in [0, V_r)$ where V_r is the number of grids in each Link SDFs and link transformations $T_{i,c}$, euclidean grid transformations $G_{i,c}$ can be computed as follows:

$$\begin{aligned} P_{xyz} &:= \begin{pmatrix} \cdots & p_{j,xyz} & \cdots \end{pmatrix} \\ \begin{pmatrix} G_{i,c} \\ 1 \end{pmatrix} &= \begin{pmatrix} R_{i,c} & \frac{\delta t_{i,c}}{e_r} \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} P_{xyz} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} R_{i,c}^T & -R_{i,c}^T \frac{\delta t_{i,c}}{e_r} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P_{xyz} \\ 1 \end{pmatrix} \end{aligned} \tag{5.4}$$

Note that $p_{i,xyz}$ are normalized in the range of $[-1, 1]$ with the voxel resolution of environment voxels $\mathbf{e_r}$. The number of columns of P_{xyz} is the total number

of voxels in the transformed Link SDFs. This batch processing of matrix-matrix multiplication is actually time-consuming because general matrix-matrix multiplications of BLAS libraries provided by vendors are optimized for square matrices and we cannot leverage the full performance of dedicated devices including GPU for tall-and-skinny matrices [180]. Instead, we use a tiny neural network f which is composed of two fully-connected layers and one ReLU activation layer to approximate and simplify this operation:

$$\begin{aligned} \delta t_{inv\ i,c} &= -R_{i,c}^{T} \frac{\delta t_{i,c}}{e_r} \\ G_{i,c} &= f(R_{i,c}) + \delta t_{inv\ i,c} \end{aligned} \tag{5.5}$$

f takes a rotation matrix and outputs euclidean grid transformations only for the specified rotation. Recent advance in deep learning provides us with a highly-optimized API for neural approximation [3]. Since the original operation is deterministic and robot-model agonistic, we can train the neural network quickly (about 10–20 mins) and reuse the pre-trained models for any type of robot once it is trained without any additional training. As described in Section 5.3.2, the maximum error of this approximation is about 1mm in our setting, which is covered by the discretization error of SDFs and therefore negligible. Therefore, the maximum error from the ground truth which derives from the total pipeline is only the discretization error: $\frac{|r_e|}{2} + \frac{|r_r|}{2}$.

5.2.2.2 Grids in Sphere instead of using Cubic Grids

Another small technique to reduce computation time is to compute transformed SDFs only for the grids in a sphere of a radius $e_r + sqrt(3(\frac{r_e}{2})^2)$ which inscribes the link. We can roughly reduce the number of grids by: $\frac{\frac{4}{3}\pi e_r^3}{(2e_r)^3} \approx 0.53$.

[3] See `https://developer.nvidia.com/cudnn`

5.3 Experiments and Results

5.3.1 System setup

All the experiments are done on a single machine, on which AMD Ryzen™ 9 4900HS and NVIDIA GeForce RTX™ 2060 with Max-Q Design are equipped for CPU and GPU. We use PyTorch and develop custom CUDA kernels for evaluation.

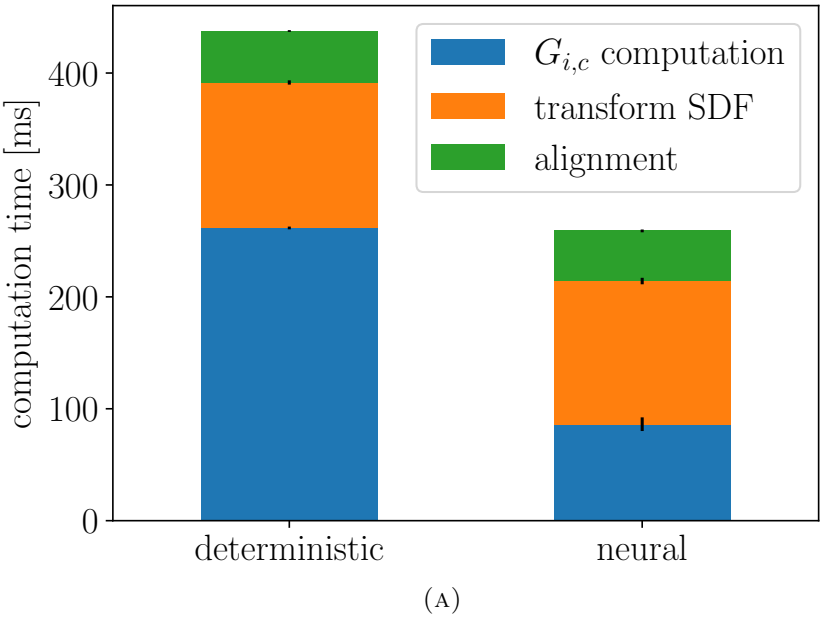
5.3.2 Precision and Speed of Neural Approximation for Euclidean Grid Transformations

First, we examine the effect of approximation of euclidean grid transformations in Fig. 5.3. In this experiment, we train the tiny neural network of 32 hidden parameters using L1 loss and Adam optimizer with a learning rate of $1e^{-4}$. We measured the time to compute euclidean grid transformations for 500 configurations of a 6 DoF robot. Deterministic transformations are operated by a function 'torch.matmul' which internally uses cuBLAS's sgemm [4].

We compare the computation speed in Fig. 5.3a. As a result, the neural approximation of euclidean transformations G is about 3.2x faster than the deterministic one, and the total SDFs computation becomes about 2x faster. According to Figure 22 of [180], ours (3.2x) exceeds the performance gain of [180] (almost 2x) from cuBLAS.

We also test approximation errors from the ground truth.We use 10^7 randomly-generated link transformations to examine the maximum error. The result is that the approximation error of the euclidean grid transformations is less than 0.0013 at the maximum. This means that, in the following experiment, considering the extent of Link SDFs e_r is set to 1.2 m, the actual error in G is $0.0013 \times e_r = 1.56$ mm which is way smaller than the grid discretization size (1 cm) and therefore negligible.

[4] `https://developer.nvidia.com/cublas`



(A)

FIGURE 5.3: Computation speed of euclidean grid approximation by a tiny neural net. See Section 5.3.2 for detail.

5.3.3 Comparison with a robot in simulation

Secondly, we compare our method with a sphere-based distance checker in simulation (Fig. 5.4). The robot loops between point A and point B while the experimenter is in close proximity to the robot and randomly moves his arms beside the robot impeding the robot's motion. The robot's motion is planned for each trajectory at runtime. The experimenter's motion is recorded by a Kinect v2, and we replay the obtained sequence of pointclouds in each experiment at the same timing. The pointcloud is converted into 4 cm voxels at runtime. We record a total time trajectory execution time for the robot to move back and forth for 6 laps over 10 trials. The protective distance d_{prot} is set to 3 cm and the extent of Link SDFs $\mathbf{e_r}$ is set to 1.2 m. The resolution of pre-computed Link SDFs is set to 1 cm. The clearance threshold is set to $sqrt((4/2)^2 \times 3) + sqrt((1/2)^2 \times 3) \approx 4.3$ cm. Our code is based on OpenRAVE [173].

At runtime, after planning a trajectory between points using an off-the-shelf RRT-based motion planner in OpenRAVE and before executing the trajectory, intermediate waypoints are sampled using TOPP-RA's automatic gridpoint suggestion feature [15]. The number of waypoints (i.e. the batch size) ranges in 300–500 depending on each trajectory. Robot SDFs are then computed with our proposed

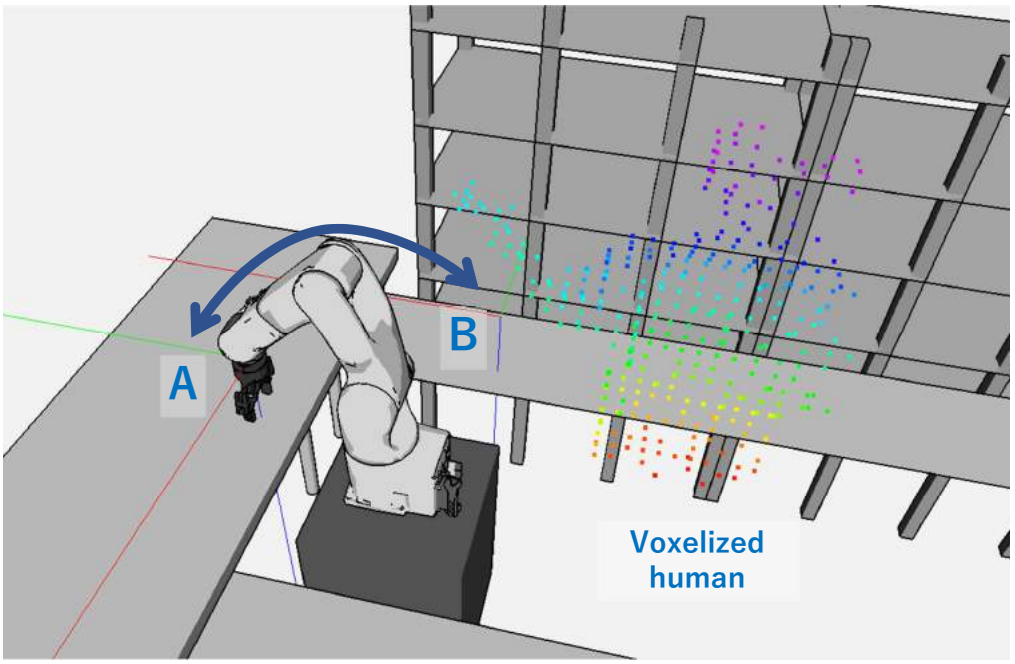


FIGURE 5.4: The experimental setup used to compare our method with a simple method which models a robot with spheres. The robot loops between point A and point B while the experimenter virtually picks up objects from a shelf aside the robot hindering the robot's motion.

method for each waypoint configuration in a parallel, batched manner. During the trajectory execution, the computed SDFs are used to retrieve the distances between a robot and obstacles for each waypoint, and time-optimal safe velocity is computed and applied to a robot at every control cycle based on [178].

To ensure safety, $\mathbf{e_r}$ needs to be large enough to capture the obstacle coming closer to a moving robot. We select the value (1.2 m) as follows: Given the joint velocity limit $v_{limit,i}$ and acceleration limit $a_{limit,i}$ for joint i, the maximum braking time t_{brake} is computed as $\max_i \frac{v_{limit,i}}{a_{limit,i}}$, which is 0.2 sec for DENSO Robot VS-060. Therefore, given the maximum velocity of obstacles v_{obs}, the system needs to capture obstacles coming closer less than $v_{obs}t_{brake} + d_{prot}$ from a robot trajectory. In our case, considering the maximum size from the center of robot links is 0.6 m, $\mathbf{e_r}$ must be larger than $1.6 \cdot 0.2 + 0.03 + 0.6 = 0.95$ m.

As a result, the robot with sphere model takes 39.53 sec to execute its entire task while ours takes 31.81 sec, which is 1.24 times faster (Fig. 5.5, Fig. 5.7). The SDFs computation takes approximately 0.2–0.3 sec per one trajectory, which is reasonable

to compute at runtime. During trajectory execution, the sphere-based checker takes 5.47 ms to batch-process distances even on GPU, while our batched distance checker requires only 0.4 ms. This is beneficial for achieving high throughput in a real-time control system (Fig. 5.5, Table 5.1). It is worth noting that our method invests 2 seconds in total at runtime in SDFs preparation for each trajectory execution and is still faster. If trajectory replanning is unnecessary and the robot follows fixed trajectories and SDFs computation can be done in advance, the total improvement is 1.44 times (from 30.51 s to 21.25 s). The comparison videos can be checked from `https://youtu.be/wO6PiOlsu-w` and `https://youtu.be/YBPpki4fGF8`. Fig. 5.6 illustrates a typical trajectory execution, logging the minimum distance to obstacles and joint velocities. A dotted horizontal line in the plot represents the protective distance. This plot shows that safety is secured by stopping the robot before collisions happen.

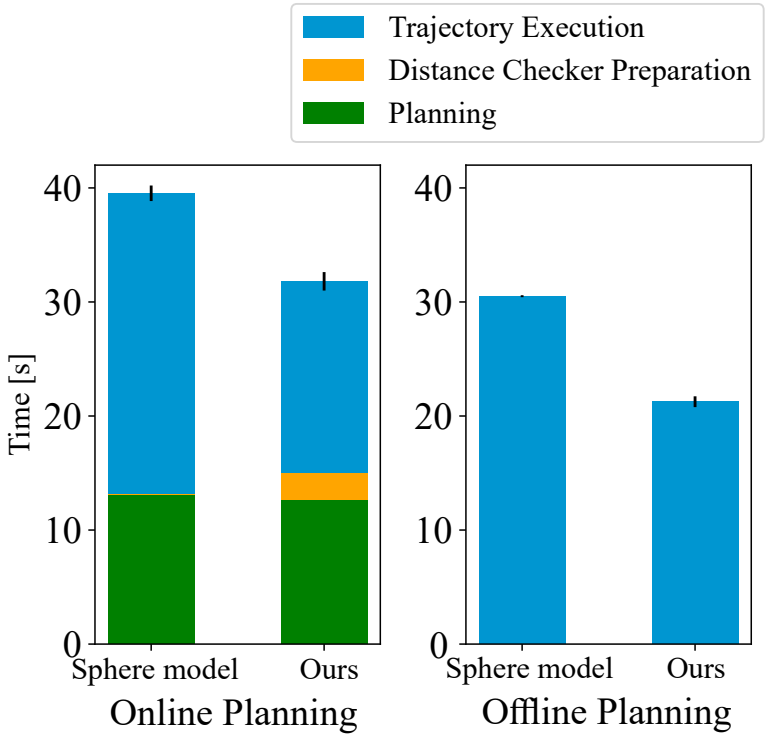


FIGURE 5.5: Comparison of execution times in a dynamic environment. Despite the additional time spent on preparing SDFs, our method exhibits a 1.24x faster total execution time compared to the simple sphere model. In the case of offline planning (i.e. trajectories are fixed and preparation is done offline), we observe a 1.44x speedup.

TABLE 5.1: Comparison of execution times per one trajectory

	Sphere model	Ours (neural approx.)
Distance Checker Preparation per 1 trajectory [s]	0.15 ± 0.012	2.34 ± 0.057
Runtime Evaluation per 1 trajectory [ms]	5.47 ± 0.096	0.391 ± 0.0014

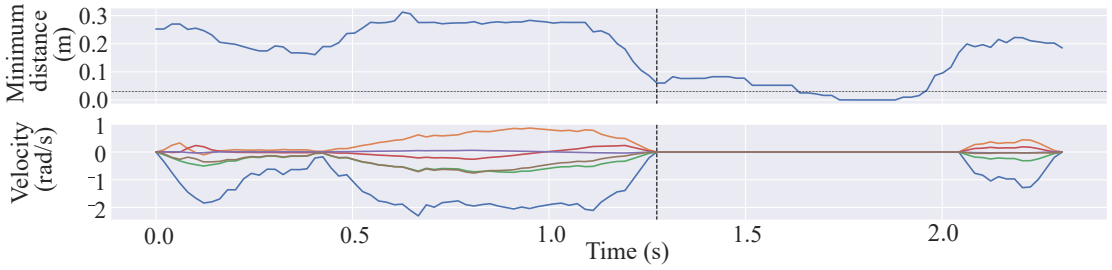
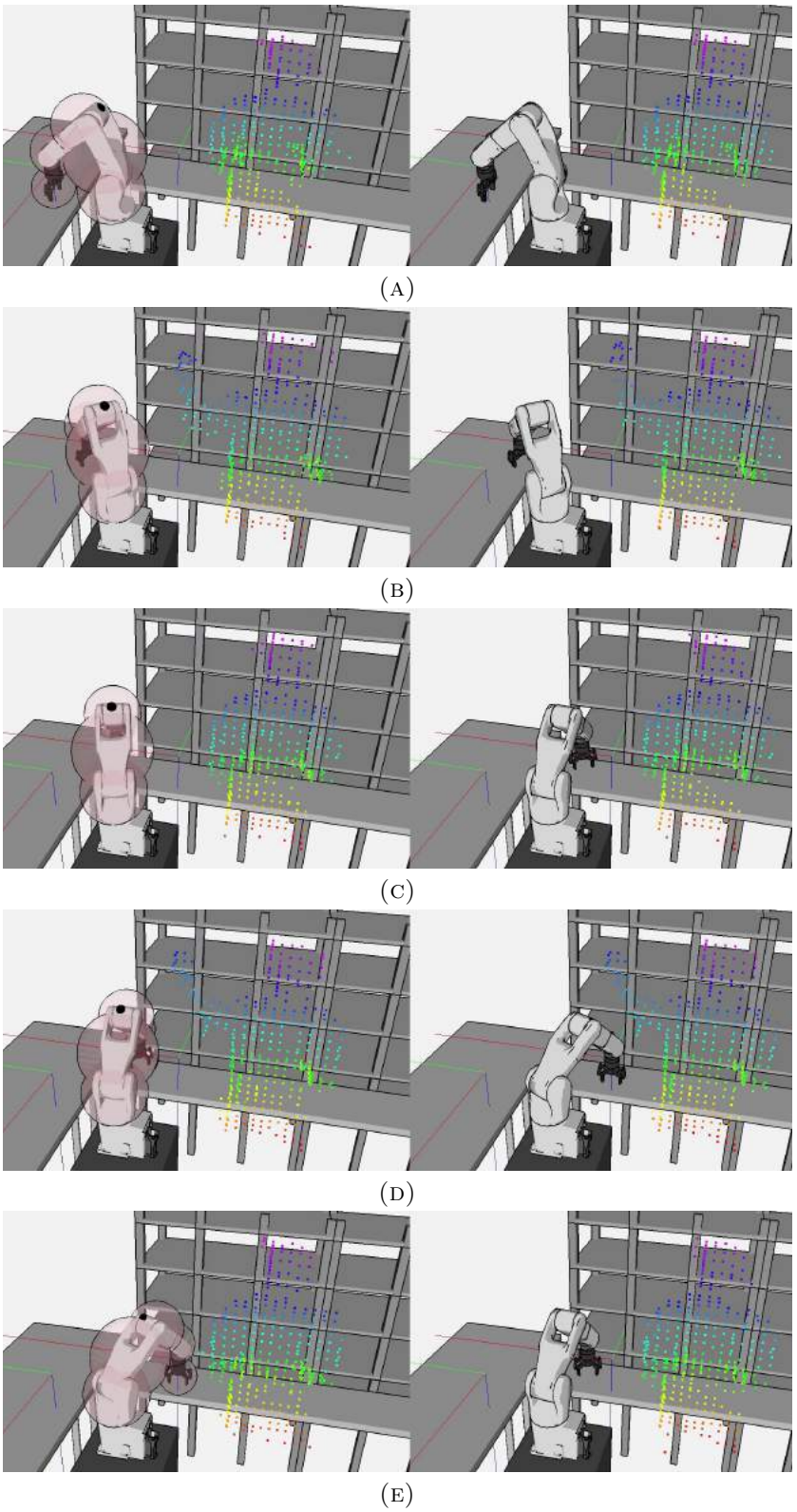


FIGURE 5.6: A result of a trajectory execution, logging Minimum distance from voxelized obstacles and Joint Velocities.

We conduct another comparison in a different setting where the robot repeatedly performs a pick-inspect-place task while the worker hinders the robot's motion behind the robot (Fig. 5.8). In this setting, the robot with sphere model takes 59.79 s to execute its entire task while ours takes 49.61 s, which is 1.21 times faster (Fig. 5.9) even investing 3.6 s in total at runtime in SDFs preparation. In the case of fixed trajectories, the total improvement is also 1.16 times (from 50.64 s to 43.52 s). The comparison videos can be checked from `https://youtu.be/W0os5xngUo4` and `https://youtu.be/Dcx55njqA5g`.

5.3.4 Real Robot Experiment

Finally, we experimentally test our algorithm with a real 6 DoF robot (Fig. 5.10). During the experiment, the robot moves between several configurations while an experimenter inhibits its motion. The environment is captured using Intel RealSense™D455 and voxelized with a resolution of 4 cm. The maximum speed of obstacles is set to 2.0 m/s, and the robot speed is limited to 80% of its capacity for the safety of the experimenter.The experiment can be viewed at `https://youtu.be/iZP_6rD34lA`.



(A)

(B)

(C)

(D)

(E)

FIGURE 5.7: Experimental comparison of our method with a simple sphere robot model. Left: Sphere model / Right: Our method. The same recording of the experimenter's pointcloud is applied to both methods for comparison. The captured video clips show the first trajectory execution. (a) The robots start to move almost simultaneously. Ours is a little slower due to SDFs computation. (b) Both robots move almost at the same speed initially and starts being hindered by the experimenter. (c) Ours accelerates earlier, (d) arrives at the destination earlier, (e) and starts returning back when the robot with the sphere model arrives. The full experiment can be viewed at `https://youtu.be/wO6PiOlsu-w`.

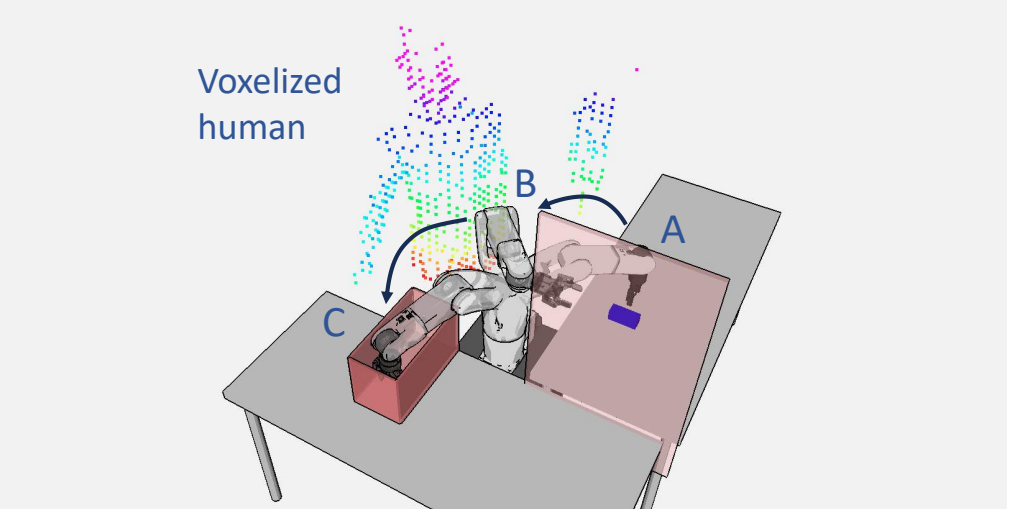


FIGURE 5.8: The second experimental setup used to compare our method with a simple method which models a robot with spheres. The robot loops between point A, B and C while the experimenter virtually hinders the robot's motion. The experimenter is captured and recorded by Kinect V2 and voxelized in 4 cm resolution.

5.4 Summary

A batched, fast, and precise distance checker has been required for truly time-optimal safe path tracking in human-robot collaborative environments. In this chapter, we propose a real-time batched distance checking method based on pre-computed link-local SDFs. Our method can check distances between a robot and obstacles along a trajectory within less than 1 millisecond on GPU, which is suitable for time-critical safety control under a collaborative situation. Additionally, to accelerate a preprocessing process, a neural approximation has been proposed, which makes the preprocessing 2x faster. Finally, we have experimentally demonstrated that our method can navigate a robot earlier than a fast but conservative geometric-primitives based distance checker in a dynamic environment.

It should be mentioned that, by introducing a greater number of smaller spheres in the simple sphere model to enhance its precision, the performance improvement of our method will diminish and the time invested in SDFs preparation may not be justified. However, as the number of spheres and occupied voxels increases, the runtime speed of the sphere-based checker grows linearly, which is critical for ensuring safe control. Indeed, during our experiments, we encountered situations where the runtime speed was insufficient for the control cycle, particularly in larger

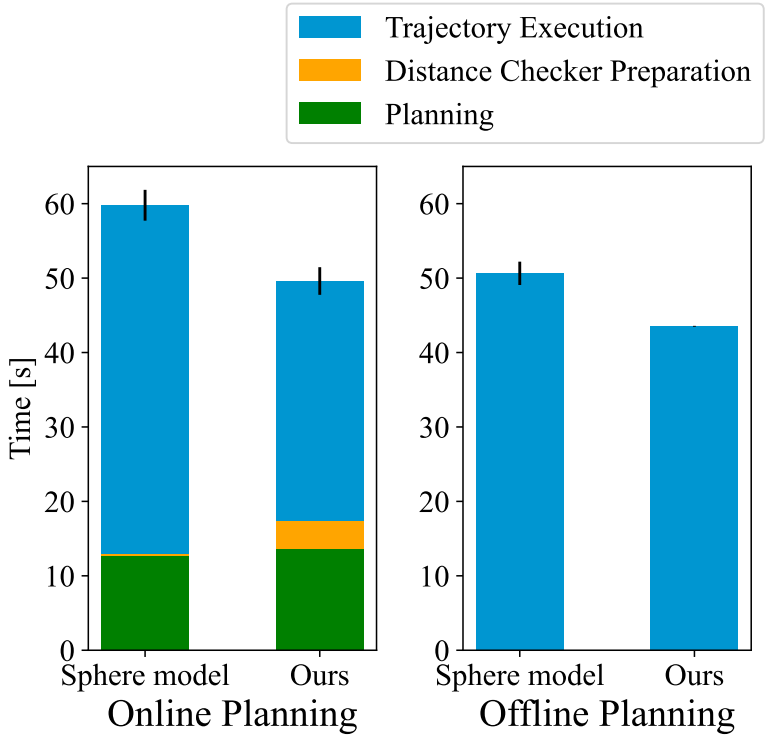
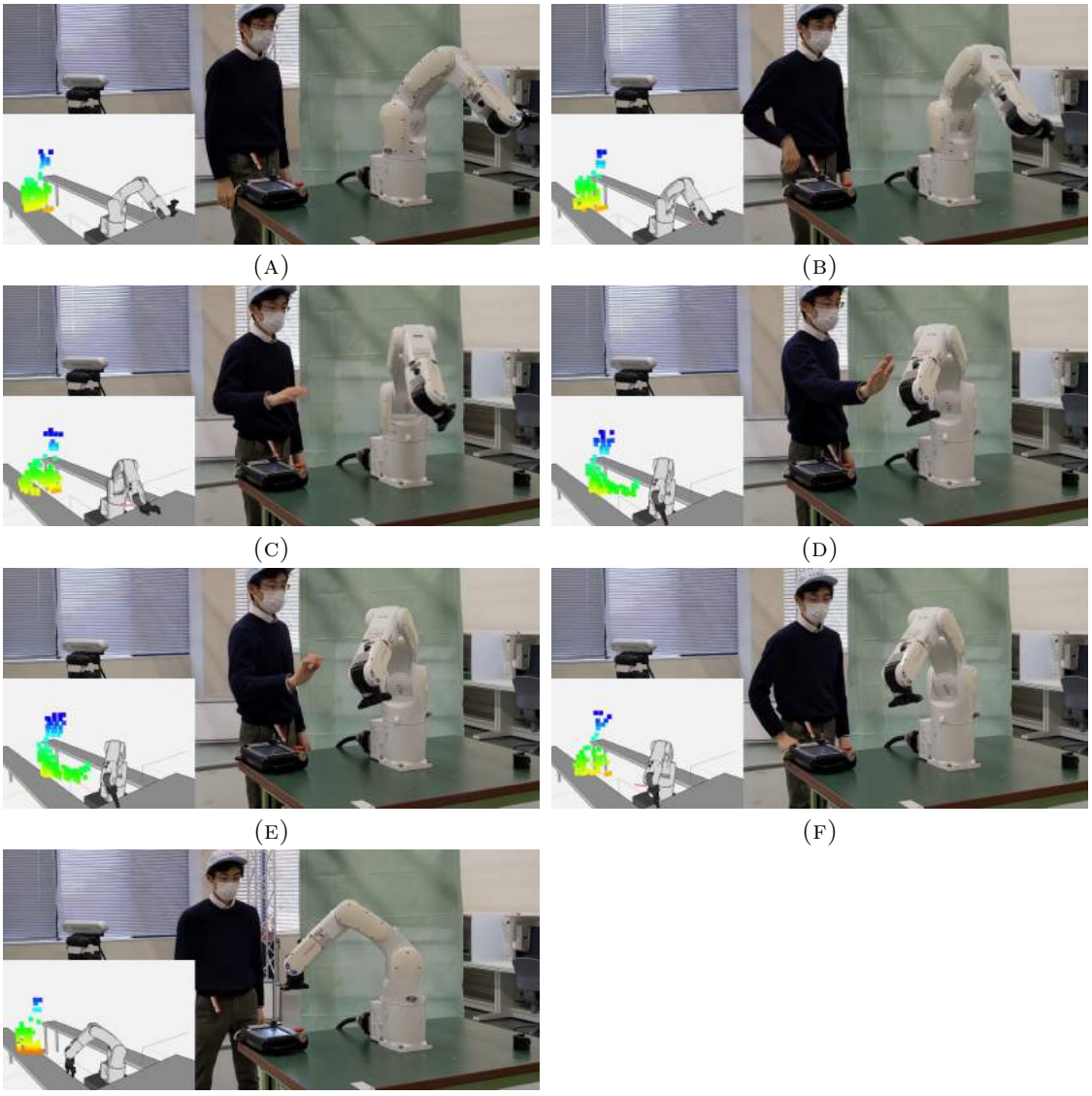


FIGURE 5.9: Another comparison of execution times in a dynamic environment. Despite the additional time spent on preparing SDFs, our method exhibits a 1.24x faster total execution time compared to the simple sphere model. In the case of offline planning (i.e. trajectories are fixed and preparation is done offline), we observe a 1.44x speedup.

and congested environments with numerous occupied voxels. On the other hand, our method maintains an advantage in terms of runtime speed. Although it scales linearly to the number of occupied voxels, it remains significantly faster since it only requires GPU memory access at runtime.

One of the limitations of our approach is its scalability in terms of space complexity. It is not well-suited for a large, finely voxelized environment due to the GPU memory consumption associated with cached SDFs. Although we do not observe a significant memory overhead inherently as the environment size increases, the GPU memory required to store SDFs increases linearly with the number of waypoints and the number of environment voxels. For instance, during the experiment described in Section 5.3.3, about 3GB GPU memory was consumed at runtime. While this issue can be mitigated by employing multiple GPUs, considering the hardware cost, further work is needed to reduce memory consumption.



(A) (B) (C) (D) (E) (F) (G)

FIGURE 5.10: Real-time safe path tracking on a physical robot. The robot moves between several start positions and a goal position while the experimenter randomly moves in the robot's path. (a) The robot starts moving towards the goal position. (b–c) The experimenter moves his hand in the robot's path. (d) The robot safely stops before the experimenter touches it. (e–f) The experimenter retreats from the robot and the robot immediately restarts moving towards the goal. (g) The robot arrives at the goal. The full experiment can be viewed at `https://youtu.be/iZP_6rD34lA`.

Chapter 6

Conclusion

6.1 Contributions

As the demand for robotic automation in a collaborative environment continues to rise, the significance of efficient and safe motion planning algorithms has become increasingly crucial. In pursuit of maximizing the productivity of collaborative robots while ensuring human safety, this thesis has presented a series of contributions to the field of real-time motion planning for human-robot collaboration.

Our contributions in this thesis are summarized as follows:

- In Chapter 3, we propose a Rapid Trajectory Smoother, which has been a missing piece in real-time motion planning, primarily to improve productivity. Existing real-time path planners lack a smoothing post-processing step, which is an essential component in sampling-based motion planning. This absence leads to planned trajectories that are jerky, resulting in inefficiency and reduced human-friendliness. Our first contribution is a Rapid Trajectory Smoother based on the shortcutting technique to address this issue. By leveraging fast clearance inference by a novel neural network (CFN), the proposed method can consistently smooth the jerky trajectories of a 6-DoF industrial robot arm within 200 ms on a single GPU, 2–3x faster than an existing method. We incorporate this suggested smoother into a full loop of Vision-Motion Planning-Execution and showcase the real-time, smooth performance of an industrial robot when faced with dynamic obstacles.

- In Chapter 4, we propose a time-optimal safe path tracking algorithm, with a dedicated focus on safety guarantee. Once the real-time path planner and the smoother plan the smooth, collision-free path, the robot needs to track the path in a time-optimal manner. However, the robot operates in a dynamic and collaborative environment, which necessitates safe movement. Specifically, the robot must stop before it collides with a human, as defined by Speed and Separation Framework in ISO/TS 15066. Driven by these contradicting objectives, we have proposed a time-optimal control policy for VPP that ensures such safe behavior. Our approach builds upon TOPP-RA [15]. Notably, we have proved that: *for any robot motion that is strictly faster than the motion computed by our policy, there exists a human motion that results in a collision with the robot in a non-stationary state.* Furthermore, we show that our policy is strictly less conservative than state-of-the-art safe robot control methods in simulation. In addition, we propose a parallelization method to significantly reduce the computation time of our pre-computation phase. The parallelization allows the entire pipeline, including pre-computation, to be executed at runtime, nearly in real-time. Experimentally we have demonstrated the application of our method in a scenario involving time-optimal, safe control of a 6-DoF industrial robot.

- In Chapter 5, we propose a batched, fast, and precise distance computation method based on precomputed link-local Signed Distance Fields(SDFs). In time-optimal safe path tracking, it is necessary to compute distances between obstacles and a robot at each waypoint along the entire executing path to evaluate *safeness* at the waypoints. In this process, no existing distance checker fulfills the demands for *true* time-optimality, that is the ability to compute distances 1. at many robot configurations, 2. in real-time, and 3. as precisely as possible. The third contribution in this thesis is a batched, fast, and precise distance computation method based on precomputed link-local SDFs. Our method can check distances for waypoints along an executing trajectory within less than 1 millisecond on GPU at runtime, making it suitable for time-critical robotic control. Additionally, we propose a neural approximation to accelerate a preprocessing process by 2x. Finally, we experimentally demonstrate that our method can navigate a 6-DoF robot earlier than a geometric-primitives based distance checker in a dynamic, collaborative environment.

Throughout this thesis, we have given special attention to the performance of our algorithms. As the topic of this thesis is closely relevant to industrial applications, the performance is pivotal for practical applicability. In each contribution, we have strategically utilized parallelization to enhance the performance, particularly by exploiting commercial and powerful GPUs. Consequently, in addition to describing and analyzing our proposed algorithms, we have developed and benchmarked GPU-accelerated implementations.

As one of the limitations of our research, we do not take several constraints into account in our smoothing and path tracking technique. In scenarios involving pick-and-place manipulations, it is common to consider constraints such as "positional constraints" of a grasped object, "cartesian speed/acceleration limits" of the end-effector, and "joint torque constraints" for optimal trajectory smoothing. Our smoother does not evaluate them, unfortunately. Our smoother assumes that collision checking is the bottleneck of trajectory smoothing, i.e., the first shortcut-candidates generation step is much faster than collision checking, which can be slow by considering those constraints. In fact, among these constraints, kinematic constraints are relatively straightforward to handle as they can be easily computed in parallel on GPU. However, dynamics constraints such as joint torque limits are challenging to consider due to their computational complexity. Developing a fast dynamics estimator will be essential to achieve fast trajectory smoothing with dynamics constraints. On the other hand, "jerk constraint" is also crucial for ensuring *human-friendliness* of a trajectory. Research has shown that minimum-jerk trajectory is psychologically acceptable [181, 182]. However, solving TOPP subject to third-order constraints such as jerk constraints is known to be difficult due to its non-convexity [183]. Consequently, jerk constraint is not handled in TOPP-RA and hence our time-optimal safe path tracking. Further research will be required for real-time, time-optimal, safe, and *human-friendly* path tracking.

By combining our proposed solutions, we can achieve online safe motion re-planning. However, the total computation time for re-planning will not be negligible. As we have reported, it takes about 300 ms to compute a collision-free smooth trajectory, 400–500 ms to calculate the velocity profile, and 250–300 ms to batch-compute precise distances for time-optimality. Hence, re-planning and switching to a new trajectory will require about 1 s in total. This time frame is significantly longer than the human's reaction time to visual stimuli (180 ms as reported in [184]).

While our proposed methods will navigate and control a robot fast and safely in nearly real time, the resulting behavior will not be sufficiently *reactive* to respond to human motion quickly. Therefore, further research is necessary to reduce the preparation time for real-time reactive motion planning.

When it comes to the deployment of our methods into real-world production environments, we need to work on their implementation within embedded robot controllers and their seamless integration with existing Robot Programming IDEs. It is imperative to engage in risk assessment and failure case analysis, addressing potential what-if scenarios such as sensor errors, sensor occlusions, communication failures, hardware/software algorithmic/systematic errors, and many others to guarantee safety. Although not covered in this thesis, evaluating the impact of these parameters and devising appropriate countermeasures is requisite for delivering a system into production. Most importantly, the resultant system must be "lucrative". We have already received weak feedback from our business department regarding an additional GPU with large memory onboard solely for speed and safety enhancements. This is because even one additional component makes the system exponentially more complex and costly, although they agree that our method itself is attractive to customers. Our future focus will be on refining our algorithm to work on cost-effective GPUs or other alternative chips, optimizing the balance between system effectiveness and cost efficiency.

6.2 Outlook for future work

In this thesis, we have worked on enhancing the productivity of collaborative robots in motion planning ensuring the safety of humans. In our work, the power of deep learning and computational parallel processing has been incorporated into the system to achieve real-time capability in motion planning and TOPP. We believe that deep learning and parallel processing will remain key technologies because they can be complementary. With deep learning, we can build a probabilistic, yet high-performing end-to-end pipeline from the experience of robots, which is apt for achieving reactivity and adaptability in robots. On the other hand, deterministic and accurate processing can be ensured by computational and mathematical parallel processing, which is suitable for guaranteeing 100% safety but is slower

compared to deep learning. Further research can be conducted to gain the characteristics of these two types of processing methods and utilize them in the best way.

There remain many interesting challenges towards a truly safe and productive robotic system. One such intriguing question is to develop an adaptive task planner. Currently, when robot users install collaborative robots, they manually program tasks and their order. However, this manual approach is problematic since, when an interruption occurs, collaborative robots stop and wait or slowly move to avoid disarranging the order of tasks, which hampers productivity. Instead, the robot may be able to pause or cancel its current task and perform other tasks, resuming its original task later once the interruption is resolved. Such task planning cannot be pre-programmed manually, since there are numerous what-if cases. Therefore, a technique for reordering the tasks quickly and adaptively can be developed, considering all of the feasibility, time-optimality, and safety of each task using machine learning and parallel computation. Additionally, the prediction of human intention can also be integrated for further productivity.

In the long term, as more complex tasks are being demanded in industrial applications, interactive and collaborative scenarios will continue to grow, where robots and humans share the workspace and work together. The contradictory objectives, safety, and productivity, will remain as long as we expect intelligence and adaptivity from robots, especially under interactive scenarios. Ultimately, our goal is to build a highly-productive, highly-reactive, and highly-adaptive robot comparable to human's abilities in the future. We believe that this would be the true machine intelligence.

In the following, we provide a brief overview of potential research avenues for a short-term perspective concerning the applications and extensions of our work.

Multi-manipulators / bimanual robot scenario – In real-world scenarios, system integrators enhance the cycle times of robotic systems by employing multiple robots or a bimanual robot in applications like robotic assembly and automotive welding [185]. Our current work can be extended and applied to address such settings for further productivity. For example, we can train the CFN with each arm and estimate clearances among the arms and obstacles for each pair. Self-collision could also be estimated by the extended CFN. However, achieving time-optimal safe path

tracking while controlling each arm independently and in a time-optimal manner will require further consideration and additional research efforts.

From coexistence to collaboration – While our method is applied to unsafe collision avoidance in a *coexistence* scenario, it can be extended to a *collaboration* scenario as well. For example, in a collaborative assembly scene, a robot can help a human assemble a product by handing over parts to the human without disturbing the human's workflow. Safe and time-optimal handover can be considered as a natural extension of our framework. For another example, in a collaborative packing and palletization scenario, a robot works alongside a human to pack products into boxes and palletize boxes onto a pallet without causing interference. Our framework can be similarly extended to such a case by considering multiple goals to put objects in a planner and finding a safe and time-optimal path to reach the best goal.In such a collaboration, trustable human motion prediction would also be required to achieve a better performance [186].

Extension to mobile robots and other areas – Our method, originally designed for fixed manipulators, has the potential for other applications, including mobile (articulated) robots and self-driving cars. Motion planning of a mobile articulated robot is more challenging than that of a fixed manipulator due to its larger dimensionality and more dynamic environment. Such an extension will require a more memory-efficient and scalable collision/distance checker. Regarding self-driving cars, it is argued that the overly conservative behavior of a self-driving car can introduce "semantic vulnerability" into the autonomous driving system, resulting in a bad user experience and unsafe incidents [187]. Time-optimal safe control of self-driving cars could be another potential application of our path-tracking method.

List of Author's Awards, Patents, and Publications[1]

Conference Proceedings

- **Shohei Fujii**, Quang-Cuong Pham. "Realtime Trajectory Smoothing with Neural Nets," in *International Conference on Robotics and Automation (ICRA)*, 7248-7254 2022. DOI: 10.1109/ICRA46639.2022.9812418.
- **Shohei Fujii**, Quang-Cuong Pham. "Time-Optimal Path Tracking with ISO Safety Guarantees," in *International Conference on Intelligent Robots and Systems (IROS)*, 2023 DOI: 10.1109/IROS55552.2023.10342287.
- **Shohei Fujii**, Quang-Cuong Pham. "Real-time Batched Distance Computation for Time-Optimal Safe Path Tracking," accepted in *International Conference on Robotics and Automation (ICRA)*, 2024.

[1]The superscript * indicates joint first authors

Bibliography

[1] International Federation of Robotics. World Robotics 2022. Technical report, International Federation of Robotics, 2022. URL https://ifr.org/downloads/press2018/2022_WR_extended_version.pdf. 1, 3

[2] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. Recommendations for implementing the strategic initiative industrie 4.0, acatech–national academy of science and engineering. *Federal Ministry of Education and Research*, 2013. 2

[3] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 2018. doi: 10.1016/j.mechatronics.2018.02.009. URL https://www.sciencedirect.com/science/article/pii/S0957415818300321. 2, 3, 9, 45

[4] ISO 3691-4:2020. Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems. Technical report, International Organization for Standardization, 2016. 2

[5] ISO/TS 15066. Robots and robotic devices collaborative robots, 2016. 2, 5, 10, 45, 51, 61, 67

[6] José Saenz, Norbert Elkmann, Olivier Gibaru, and Pedro Neto. Survey of methods for design of collaborative robotics applications- why safety is a barrier to more widespread robotics uptake. In *Proceedings of the 2018 4th International Conference on Mechatronics and Robotics Engineering*, ICMRE 2018, page 95–101. Association for Computing Machinery, 2018. ISBN 9781450363655. doi: 10.1145/3191477.3191507. URL https://doi.org/10.1145/3191477.3191507. 3

[7] Atieh Hanna, Simon Larsson, Per-Lage Götvall, and Kristofer Bengtsson. Deliberative safety for industrial intelligent human–robot collaboration: Regulatory challenges and solutions for taking the next step towards industry 4.0. *Robotics and Computer-Integrated Manufacturing*, 78:102386, 2022. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2022.102386. URL https://www.sciencedirect.com/science/article/pii/S0736584522000734. 3

[8] L. Wang, R. Gao, J. Váncza, J. Krüger, X.V. Wang, S. Makris, and G. Chryssolouris. Symbiotic human-robot collaborative assembly. *CIRP Annals*, 68

(2):701–726, 2019. ISSN 0007-8506. doi: https://doi.org/10.1016/j.cirp.2019.05.002. URL https://www.sciencedirect.com/science/article/pii/S0007850619301593. 3

[9] Quang-Cuong Pham and Yoshihiko Nakamura. A new trajectory deformation algorithm based on affine transformations. *IEEE Transactions on Robotics*, 31(4):1054–1063, 2015. 4, 18, 32

[10] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985. doi: 10.1177/027836498500400301. URL https://doi.org/10.1177/027836498500400301. 5

[11] E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 1, pages 43–49 vol.1, 2001. doi: 10.1109/ACC.2001.945511. 6

[12] Kris Hauser. On responsiveness, safety, and completeness in real-time motion planning. *Autonomous Robots*, 32(1):35–48, September 2011. doi: 10.1007/s10514-011-9254-z. URL https://doi.org/10.1007/s10514-011-9254-z. 6

[13] Sean Murray, Will Floyd-Jones, Ying Qi, Daniel Sorin, and George Konidaris. Robot motion planning on a chip. In *Proceedings of Robotics: Science and Systems*, June 2016. doi: 10.15607/RSS.2016.XII.004. 6, 17, 24, 31

[14] Jia Pan and Dinesh Manocha. GPU-based parallel collision detection for fast motion planning. *The International Journal of Robotics Research*, 31(2):187–200, 2012. doi: 10.1177/0278364911429335. URL https://doi.org/10.1177/0278364911429335. 6, 17, 31

[15] Hung Pham and Quang-Cuong Pham. A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, 34(3):645–659, 2018. doi: 10.1109/TRO.2018.2819195. 6, 20, 46, 48, 50, 51, 52, 57, 74, 84

[16] J. Pan, S. Chitta, and D. Manocha. FCL: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866, 2012. doi: 10.1109/ICRA.2012.6225337. 7, 26, 35, 39, 40, 68

[17] Jochen Heinzmann and Alexander ZelinskyResearch. The essential components of human-friendly robot systems. pages 43–51, 1999. 9

[18] Dana Kulić and Elizabeth Croft. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22(2):149–164, Feb 2007. ISSN 1573-7527. doi: 10.1007/s10514-006-9009-4. URL https://doi.org/10.1007/s10514-006-9009-4.

[19] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca, and Gerd Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363, 2008. doi: 10.1109/IROS.2008.4650764. 9, 10, 19

[20] Eloise Matheson, Riccardo Minto, Emanuele G. G. Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: A review. *Robotics*, 8(4), 2019. ISSN 2218-6581. doi: 10.3390/robotics8040100. URL `https://www.mdpi.com/2218-6581/8/4/100`. 9

[21] Japanese Standards Association. Jis b 84330-1, robots and robotic devices collaborative robots, 2016. 10

[22] D. Kulic and E.A. Croft. Real-time safety for human - robot interaction. In *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 719–724, 2005. doi: 10.1109/ICAR.2005.1507488. 10

[23] A. de Luca and R. Mattone. Sensorless Robot Collision Detection and Hybrid Force/Motion Control. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 999–1004, 2005. doi: 10.1109/ROBOT.2005.1570247. 10

[24] Emanuele Magrini and Alessandro De Luca. Hybrid force/velocity control for physical human-robot collaboration tasks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2016. doi: 10.1109/iros.2016.7759151. URL `https://doi.org/10.1109/iros.2016.7759151`. 10

[25] Matteo Ragaglia, Andrea Maria Zanchettin, and Paolo Rocco. Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. *Mechatronics*, 55:267–281, 2018. ISSN 0957-4158. doi: https://doi.org/10.1016/j.mechatronics.2017.12.009. URL `https://www.sciencedirect.com/science/article/pii/S0957415817301861`. 10, 13, 15, 25

[26] J. A. Corrales, G. J. García Gómez, F. Torres, and V. Perdereau. Cooperative tasks between humans and robots in industrial environments. *International Journal of Advanced Robotic Systems*, 9(3):94, 2012. doi: 10.5772/50988. URL `https://doi.org/10.5772/50988`. 11

[27] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985. doi: 10.1109/ROBOT.1985.1087247. 12

[28] Oliver Brock and Oussama Khatib. Elastic strips: A framework for integrated planning and execution. In *Experimental Robotics VI*, pages 329–338, London, 2000. Springer London. ISBN 978-1-84628-541-7. doi: 10.1007/BFb0119411. 12, 18

[29] James Kuffner and Jing Xiao. *Motion for Manipulation Tasks*, pages 897–930. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32552-1. doi: 10.1007/978-3-319-32552-1_36. URL https://doi.org/10.1007/978-3-319-32552-1_36. 12

[30] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1–6, 1987. doi: 10.1109/ROBOT.1987.1088038. 12

[31] Xiaoping Yun and Ko-Cheng Tan. A wall-following method for escaping local minima in potential field based motion planning. In *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*, pages 421–426, 1997. doi: 10.1109/ICAR.1997.620216.

[32] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5): 501–518, 1992. doi: 10.1109/70.163777. 12

[33] K Sato. Collision avoidance in multi-dimensional space using laplace potential. In *Proc. 15th Conf. Robotics Soc. Jpn*, pages 155–156, 1987. 12

[34] J.-O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3): 338–349, 1992. doi: 10.1109/70.143352. 12

[35] Kyle Hollins Wray, Dirk Ruiken, Roderic A. Grupen, and Shlomo Zilberstein. Log-space harmonic function path planning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1511–1516. IEEE, 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759245. URL http://ieeexplore.ieee.org/document/7759245/. 12

[36] Renjie Chen, Craig Gotsman, and Kai Hormann. Path planning with divergence-based distance functions. *Computer Aided Geometric Design*, 66: 52–74, 2018. ISSN 0167-8396. doi: https://doi.org/10.1016/j.cagd.2018.09.002. URL https://www.sciencedirect.com/science/article/pii/S0167839618301067. 12

[37] Jim Mainprice, Nathan Ratliff, and Stefan Schaal. Warping the workspace geometry with electric potentials for motion optimization of manipulation tasks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3156–3163, 2016. doi: 10.1109/IROS.2016.7759488. 12

[38] Gowtham Garimella, Matthew Sheckells, and Marin Kobilarov. A stabilizing gyroscopic obstacle avoidance controller for underactuated systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5010–5016, 2016. doi: 10.1109/CDC.2016.7799035. 12

[39] Marvin Becker, Torsten Lilge, Matthias A. Müller, and Sami Haddadin. Circular fields and predictive multi-agents for online global trajectory planning. *IEEE Robotics and Automation Letters*, 6(2):2618–2625, 2021. doi: 10.1109/LRA.2021.3061997. 12

[40] Marvin Becker, Johannes Köhler, Sami Haddadin, and Matthias A. Müller. Motion planning using reactive circular fields: A 2d analysis of collision avoidance and goal convergence. *CoRR*, abs/2210.16106, 2022. doi: 10.48550/arXiv.2210.16106. URL `https://doi.org/10.48550/arXiv.2210.16106`.

[41] Marvin Becker, Philipp Caspers, Tom Hattendorf, Torsten Lilge, Sami Haddadin, and Matthias A. Müller. Informed circular fields for global reactive obstacle avoidance of robotic manipulators, 2022. 12

[42] Bakir Lacevic and Paolo Rocco. Kinetostatic danger field - a novel safety assessment for human-robot interaction. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2169–2174, 2010. doi: 10.1109/IROS.2010.5649124. 12

[43] Bakir Lacevic, Paolo Rocco, and Andrea Maria Zanchettin. Safety assessment and control of robotic manipulators using danger field. *IEEE Transactions on Robotics*, 29(5):1257–1270, 2013. doi: 10.1109/TRO.2013.2271097. 12

[44] Matteo Parigi Polverini, Andrea Maria Zanchettin, and Paolo Rocco. Real-time collision avoidance in human-robot interaction based on kinetostatic safety field. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4136–4141. IEEE, 2014. ISBN 978-1-4799-6934-0. doi: 10.1109/IROS.2014.6943145. URL `https://ieeexplore.ieee.org/document/6943145/`. 12

[45] Jinwook Huh, Volkan Isler, and Daniel D. Lee. Cost-to-go function generating networks for high dimensional motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8480–8486, 2021. doi: 10.1109/ICRA48506.2021.9561672. 12

[46] Alessandro De Luca and Fabrizio Flacco. Integrated control for phri: Collision avoidance, detection, reaction and collaboration. In *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 288–295, 2012. doi: 10.1109/BioRob.2012.6290917. 13, 22

[47] Daniel Rakita, Haochen Shi, Bilge Mutlu, and Michael Gleicher. CollisionIK: A per-instant pose optimization method for generating robot motions with environment collision avoidance. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9995–10001, 2021. doi: 10.1109/ICRA48506.2021.9561505. 13

[48] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion. In *Proceedings of*

Robotics: Science and Systems, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.043. 13

[49] Kelly Merckaert, Bryan Convens, Chi ju Wu, Alessandro Roncone, Marco M. Nicotra, and Bram Vanderborght. Real-time motion control of robotic manipulators for safe human–robot coexistence. *Robotics and Computer-Integrated Manufacturing*, 73:102223, 2022. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2021.102223. URL `https://www.sciencedirect.com/science/article/pii/S0736584521001022`. 13, 22

[50] Paul Bosscher and Daniel Hedman. Real-time collision avoidance algorithm for robotic manipulators. In *2009 IEEE International Conference on Technologies for Practical Robot Applications*, pages 113–122. IEEE, 2009. ISBN 978-1-4244-4991-0. doi: 10.1109/TEPRA.2009.5339635. URL `http://ieeexplore.ieee.org/document/5339635/`. 13

[51] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 13

[52] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL `https://www.science.org/doi/abs/10.1126/scirobotics.abc5986`. 13

[53] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. doi: 10.1126/scirobotics.abk2822. URL `https://www.science.org/doi/abs/10.1126/scirobotics.abk2822`. 13

[54] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439. 13, 17, 31

[55] Jung Jun Park, Ji Hun Kim, and Jae Bok Song. Path planning for a robot manipulator based on probabilistic roadmap and reinforcement learning. *International Journal of Control, Automation and Systems*, 5(6):674–680, December 2007. ISSN 1598-6446. 13

[56] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992. doi: 10.1007/bf00992698. URL `https://doi.org/10.1007/bf00992698`. 13

[57] Robin Strudel, Ricardo Garcia Pinel, Justin Carpentier, Jean-Paul Laumond, Ivan Laptev, and Cordelia Schmid. Learning obstacle representations for neural motion planning. In Jens Kober, Fabio Ramos, and Claire Tomlin,

editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 355–364. PMLR, 16–18 Nov 2021. URL `https://proceedings.mlr.press/v155/strudel21a.html`. 13

[58] Liam Schramm and Abdeslam Boularias. Learning-guided exploration for efficient sampling-based motion planning in high dimensions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4429–4435, 2022. doi: 10.1109/ICRA46639.2022.9812184. 13

[59] Bianca Sangiovanni, Angelo Rendiniello, Gian Paolo Incremona, Antonella Ferrara, and Marco Piastra. Deep reinforcement learning for collision avoidance of robotic manipulators. In *2018 European Control Conference (ECC)*, pages 2063–2068, 2018. doi: 10.23919/ECC.2018.8550363. 13

[60] Quan Liu, Zhihao Liu, Bo Xiong, Wenjun Xu, and Yang Liu. Deep reinforcement learning-based safe interaction for industrial human-robot collaboration using intrinsic reward function. *Advanced Engineering Informatics*, 49:101360, 2021. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2021.101360. URL `https://www.sciencedirect.com/science/article/pii/S1474034621001130`. 14

[61] Yue Shen, Qingxuan Jia, Zeyuan Huang, Ruiquan Wang, Junting Fei, and Gang Chen. Reinforcement learning-based reactive obstacle avoidance method for redundant manipulators. *Entropy*, 24(2), 2022. ISSN 1099-4300. doi: 10.3390/e24020279. URL `https://www.mdpi.com/1099-4300/24/2/279`. 14

[62] Mohamed El-Shamouty, Xinyang Wu, Shanqi Yang, Marcel Albus, and Marco F. Huber. Towards safe human-robot collaboration using deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4899–4905, 2020. doi: 10.1109/ICRA40945.2020.9196924. 14

[63] Xuan Zhao, Tingxiang Fan, Yanwen Li, Yu Zheng, and Jia Pan. An efficient and responsive robot motion controller for safe human-robot collaboration. *IEEE Robotics and Automation Letters*, 6(3):6068–6075, 2021. doi: 10.1109/LRA.2021.3088091. 14

[64] Jakob Thumm and Matthias Althoff. Provably safe deep reinforcement learning for robotic manipulation in human environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6344–6350, 2022. doi: 10.1109/ICRA46639.2022.9811698. 14

[65] Lars Grüne and Jürgen Pannek. Nonlinear Model Predictive Control. In Lars Grüne and Jürgen Pannek, editors, *Nonlinear Model Predictive Control: Theory and Algorithms*, Communications and Control Engineering, pages 43–66. Springer, 2011. ISBN 978-0-85729-501-9. doi: 10.1007/978-0-85729-501-9_3. URL `https://doi.org/10.1007/978-0-85729-501-9_3`. 14

[66] Sebastien Kleff, Avadesh Meduri, Rohan Budhiraja, Nicolas Mansard, and Ludovic Righetti. High-frequency nonlinear model predictive control of a manipulator. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7330–7336, 2021. doi: 10.1109/ICRA48506.2021.9560990. 14

[67] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966. doi: 10.1080/00207176608921369. URL `https://doi.org/10.1080/00207176608921369`. 14

[68] Siqi Hu, Edwin Babaians, Mojtaba Karimi, and Eckehard Steinbach. NMPC-MP: Real-time Nonlinear Model Predictive Control for Safe Motion Planning in Manipulator Teleoperation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8309–8316. IEEE, 2021. ISBN 978-1-66541-714-3. doi: 10.1109/IROS51168.2021.9636802. URL `https://ieeexplore.ieee.org/document/9636802/`. 15

[69] Julian Nubert, Johannes Köhler, Vincent Berenz, Frank Allgöwer, and Sebastian Trimpe. Safe and fast tracking on a robot manipulator: Robust mpc and neural network control. *IEEE Robotics and Automation Letters*, 5(2): 3050–3057, 2020. doi: 10.1109/LRA.2020.2975727. 15

[70] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4. 15

[71] Christoph Rösmann, Maximilian Krämer, Artemi Makarow, Frank Hoffmann, and Torsten Bertram. Exploiting sparse structures in nonlinear model predictive control with hypergraphs. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1332–1337, 2018. doi: 10.1109/AIM.2018.8452378. 15

[72] Maximilian Krämer, Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Model predictive control of a collaborative manipulator considering dynamic obstacles. *Optimal Control Applications and Methods*, 41 (4):1211–1232, 2020. doi: https://doi.org/10.1002/oca.2599. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.2599`. 15

[73] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202. 15

[74] Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht,

Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 277–286. PMLR, 10–11 Jun 2020. URL `https://proceedings.mlr.press/v120/bharadhwaj20a.html`. 15

[75] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 750–759. PMLR, 2022. URL `https://proceedings.mlr.press/v164/bhardwaj22a.html`. 15

[76] S. Joe Qin and Thomas A. Badgwell. An Overview of Nonlinear Model Predictive Control Applications. In Frank Allgöwer and Alex Zheng, editors, *Nonlinear Model Predictive Control*, Progress in Systems and Control Theory, pages 369–392. Birkhäuser, 2000. ISBN 978-3-0348-8407-5. doi: 10.1007/978-3-0348-8407-5_21. 15

[77] Chi-Shen Tsai, Jwu-Sheng Hu, and Masayoshi Tomizuka. Ensuring safety in human-robot coexistence environment. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4191–4196, 2014. doi: 10.1109/IROS.2014.6943153. 15

[78] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. *CoRR*, abs/1903.11199, 2019. URL `http://arxiv.org/abs/1903.11199`. 16

[79] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017. doi: 10.1109/TAC.2016.2638961. 16

[80] Xuda Ding, Han Wang, Yi Ren, Yu Zheng, Cailian Chen, and Jianping He. Safety-critical optimal control for robotic manipulators in a cluttered environment, 2022. 16

[81] Andrew Singletary, William Guffey, Tamas G. Molnar, Ryan Sinnet, and Aaron D. Ames. Safety-critical manipulation for collision-free food preparation, 2022. 16

[82] Kamal Kant and Steven W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986. doi: 10.1177/027836498600500304. URL `https://doi.org/10.1177/027836498600500304`. 16

[83] Z. Shiller and S. Dubowsky. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797, 1991. doi: 10.1109/70.105387. 16

[84] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998. 17, 31

[85] Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009. ISBN 978-1-4244-2788-8. doi: 10.1109/ROBOT.2009.5152817. URL `http://ieeexplore.ieee.org/document/5152817/`. 17, 18, 27

[86] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion Planning with Sequential Convex Optimization and Convex Collision Checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364914528132. URL `http://journals.sagepub.com/doi/10.1177/0278364914528132`. 17

[87] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985. doi: 10.1177/027836498500400301. URL `https://doi.org/10.1177/027836498500400301`. 17, 19, 20

[88] Quang-Cuong Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Transactions on Robotics*, 30(6):1533–1540, 2014. doi: 10.1109/TRO.2014.2351113. 17, 19, 20

[89] Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. RT-RRT*: A real-time path planning algorithm based on RRT*. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, pages 113–118. Association for Computing Machinery, 2015. ISBN 978-1-4503-3991-9. doi: 10.1145/2822013.2822036. URL `https://doi.org/10.1145/2822013.2822036`. 17

[90] Jia Pan, Christian Lauterbach, and Dinesh Manocha. g-Planner: Real-time motion planning and global navigation using GPUs. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 17, 31

[91] Chonhyon Park, Jia Pan, and Dinesh Manocha. Parallel Motion Planning Using Poisson-Disk Sampling. *IEEE Transactions on Robotics*, 33(2):359–371, 2017. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2016.2632160. URL `http://ieeexplore.ieee.org/document/7790881/`. 17

[92] Peter Leven and Seth Hutchinson. A framework for real-time path planning in changing environments. *The International Journal of Robotics Research*, 21(12):999–1030, 2002. doi: 10.1177/027836490202101201. URL `https://doi.org/10.1177/027836490202101201`. 17

[93] Yiming Yang, Wolfgang Merkt, Vladimir Ivan, Zhibin Li, and Sethu Vijayakumar. HDRM: A Resolution Complete Dynamic Roadmap for Real-Time Motion Planning in Complex Scenes. *IEEE Robotics and Automation Letters*, 3(1):551–558, 2018. doi: 10.1109/LRA.2017.2773669. 17

[94] Shiqi Lian, Yinhe Han, Xiaoming Chen, Ying Wang, and Hang Xiao. Dadu-P: A scalable accelerator for robot motion planning in a dynamic environment. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6, 2018. doi: 10.1109/DAC.2018.8465785. 18

[95] Yinhe Han, Yuxin Yang, Xiaoming Chen, and Shiqi Lian. Dadu series: Fast and efficient robot accelerators. In *Proceedings of the 39th International Conference on Computer-Aided Design*, ICCAD '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380263. doi: 10.1145/3400302.3415759. URL `https://doi.org/10.1145/3400302.3415759`. 18

[96] Roland Geraerts and Mark H Overmars. Creating high-quality paths for motion planning. *The international journal of robotics research*, 26(8):845–863, 2007. 18, 32

[97] Kris Hauser and Victor Ng-Thow-Hing. Fast Smoothing of Manipulator Trajectories Using Optimal Bounded-Acceleration Shortcuts. In *2010 IEEE International Conference on Robotics and Automation*, pages 2493–2498. IEEE, 2010. ISBN 978-1-4244-5038-1. doi: 10.1109/ROBOT.2010.5509683. URL `http://ieeexplore.ieee.org/document/5509683/`. 18, 32, 39, 40, 41

[98] Ran Zhao and Daniel Sidobre. Trajectory smoothing using jerk bounded shortcuts for service manipulator robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4929–4934, 2015. doi: 10.1109/IROS.2015.7354070. 18

[99] Jia Pan, Liangjun Zhang, and Dinesh Manocha. Collision-free and smooth trajectory computation in cluttered environments. *The International Journal of Robotics Research*, 31(10):1155–1175, 2012. 18

[100] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7): 846–894, 2011. doi: 10.1177/0278364911406761. URL `https://doi.org/10.1177/0278364911406761`. 18

[101] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, 2011. doi: 10.1109/ICRA.2011.5980479. 18

[102] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011. doi: 10.1109/ICRA.2011.5980280. 18

[103] Chonhyon Park, Jia Pan, and Dinesh Manocha. Real-time optimization-based planning in dynamic environments using GPUs. In *IEEE International Conference on Robotics and Automation*, pages 4090–4097. IEEE, 2013. ISBN 978-1-4673-5643-5 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6631154. URL `http://ieeexplore.ieee.org/document/6631154/`. 19

[104] Alan Kuntz, Chris Bowen, and Ron Alterovitz. Fast Anytime Motion Planning in Point Clouds by Interleaving Sampling and Interior Point Optimization. In Nancy M. Amato, Greg Hager, Shawna Thomas, and Miguel Torres-Torriti, editors, *Robotics Research*, volume 10, pages 929–945. Springer International Publishing, 2017. ISBN 978-3-030-28618-7 978-3-030-28619-4. doi: 10.1007/978-3-030-28619-4_63. URL `http://link.springer.com/10.1007/978-3-030-28619-4_63`. 19

[105] Siyu Dai, Matthew Orton, Shawn Schaffert, Andreas Hofmann, and Brian Williams. Improving Trajectory Optimization Using a Roadmap Framework. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8674–8681, 2018. doi: 10.1109/IROS.2018.8594274. 19

[106] Andrea Maria Zanchettin, Nicola Maria Ceriani, Paolo Rocco, Hao Ding, and Björn Matthias. Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2):882–893, 2016. doi: 10.1109/TASE.2015.2412256. 19, 20, 22, 26, 45, 59

[107] Matteo Ragaglia, Andrea Maria Zanchettin, and Paolo Rocco. Safety-aware trajectory scaling for human-robot collaboration with prediction of human occupancy. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 85–90, 2015. doi: 10.1109/ICAR.2015.7251438. 19, 20, 25, 26

[108] Christoffer Sloth and Henrik Gordon Petersen. Computation of Safe Path Velocity for Collaborative Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6142–6148, 2018. doi: 10.1109/IROS.2018.8594217. 20

[109] Sebastian Herbster, Roland Behrens, and Norbert Elkmann. A New Approach to Estimate the Apparent Mass of Collaborative Robot Manipulators. In Bruno Siciliano, Cecilia Laschi, and Oussama Khatib, editors, *Experimental Robotics*, pages 211–221. Springer International Publishing, 2021. ISBN 978-3-030-71151-1. 20

[110] Niccolò Lucci, Bakir Lacevic, Andrea Maria Zanchettin, and Paolo Rocco. Combining speed and separation monitoring with power and force limiting for safe collaborative robotics applications. *IEEE Robotics and Automation Letters*, 5(4):6121–6128, 2020. doi: 10.1109/LRA.2020.3010211. 20, 26

[111] Zvi Shiller and Hsueh-Hen Lu. Computation of Path Constrained Time Optimal Motions With Dynamic Singularities. *Journal of Dynamic Systems, Measurement, and Control*, 114(1):34–40, 03 1992. ISSN 0022-0434. doi: 10.1115/1.2896505. URL `https://doi.org/10.1115/1.2896505`. 20

[112] Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. Practical time-optimal trajectory planning for robots: a convex optimization approach. *IEEE Transactions on Automatic Control*, 14:28, 2008. 20

[113] Kris Hauser. Fast interpolation and time-optimization with contact. *The International Journal of Robotics Research*, 33(9):1231–1250, 2014. doi: 10.1177/0278364914527855. URL `https://doi.org/10.1177/0278364914527855`. 20

[114] Joshua Bialkowski, Sertac Karaman, and Emilio Frazzoli. Massively parallelizing the rrt and the rrt. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3513–3518, 2011. doi: 10.1109/IROS.2011.6095053. 21, 23

[115] S. Robla-Gómez, Victor M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria. Working together: A review on safe human-robot collaboration in industrial environments. *IEEE Access*, 5: 26754–26773, 2017. doi: 10.1109/ACCESS.2017.2773127. 22, 23

[116] Thomas Schlegl, Torsten Kröger, Andre Gaschler, Oussama Khatib, and Hubert Zangl. Virtual whiskers — highly responsive robot collision avoidance. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5373–5379, 2013. doi: 10.1109/IROS.2013.6697134. 22

[117] Nicola Maria Ceriani, Giovanni Buizza Avanzini, Andrea Maria Zanchettin, Luca Bascetta, and Paolo Rocco. Optimal placement of spots in distributed proximity sensors for safe human-robot interaction. In *2013 IEEE International Conference on Robotics and Automation*, pages 5858–5863, May 2013. doi: 10.1109/ICRA.2013.6631420. 22

[118] Yitao Ding, Felix Wilhelm, Leonhard Faulhammer, and Ulrike Thomas. With proximity servoing towards safe human-robot-interaction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4907–4912, Nov 2019. doi: 10.1109/IROS40897.2019.8968438. 22

[119] Lucian Balan and Gary M. Bone. Real-time 3d collision avoidance method for safe human and robot coexistence. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 276–282, 2006. doi: 10.1109/IROS.2006.282068. 22

[120] Aaron Pereira and Matthias Althoff. Calculating human reachable occupancy for guaranteed collision-free planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4473–4480, 2017. doi: 10.1109/IROS.2017.8206314. 22, 25

[121] Jeffrey Too Chuan Tan, Feng Duan, Ye Zhang, Kei Watanabe, Ryu Kato, and Tamio Arai. Human-robot collaboration in cellular manufacturing: Design and development. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 29–34, Oct 2009. doi: 10.1109/IROS.2009.5354155. 22

[122] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, Oct 2013. ISSN 2168-2275. doi: 10.1109/TCYB.2013.2265378. 22

[123] Andreas Hermann, Sebastian Klemm, Zhixing Xue, Arne Roennau, and Rüdiger Dillmann. Gpu-based real-time collision detection for motion execution in mobile manipulation planning. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–7, 2013. doi: 10.1109/ICAR.2013.6766473. 22, 23, 71

[124] Changliu Liu and Masayoshi Tomizuka. Algorithmic safety measures for intelligent industrial co-robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3095–3102, 2016. doi: 10.1109/ICRA.2016.7487476. 22, 27

[125] Joyce Eduardo Taboada Diaz, Ronald Boss, Peter Kyberd, Ed Norman Biden, Carlos Diaz Novo, and Maylin Hernández Ricardo. Testing the Microsoft kinect skeletal tracking accuracy under varying external factors. *MOJ Applied Bionics and Biomechanics*, 6(1):7–11, July 2022. doi: 10.15406/mojabb.2022.06.00160. URL `https://doi.org/10.15406/mojabb.2022.06.00160`. 23

[126] Ming Lin and Stefan Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces*, volume 1, pages 602–608, 1998. 23

[127] C. Lauterbach, Q. Mo, and D. Manocha. gProximity: Hierarchical GPU-based Operations for Collision and Distance Queries. *Computer Graphics Forum*, 29(2):419–428, 2010. doi: https://doi.org/10.1111/j.1467-8659.2009.01611.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01611.x`. 23

[128] Jia Pan and Dinesh Manocha. Gpu-based parallel collision detection for fast motion planning. *The International Journal of Robotics Research*, 31(2):187–200, 2012. doi: 10.1177/0278364911429335. URL `https://doi.org/10.1177/0278364911429335`.

[129] Louis Montaut, Quentin Le Lidec, Vladimir Petrik, Josef Sivic, and Justin Carpentier. Collision detection accelerated: An optimization perspective. In *Robotics: Science and Systems*, 2022. 23

[130] Floyd M. Chitalu, Christophe Dubach, and Taku Komura. Bulk-synchronous parallel simultaneous bvh traversal for collision detection on gpus. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357050. doi: 10.1145/3190834.3190848. URL `https://doi.org/10.1145/3190834.3190848`. 23

[131] Quentin Avril, Valérie Gouranton, and Bruno Arnaldi. Synchronization-Free Parallel Collision Detection Pipeline. In *(20th International Conference on Artificial Reality and Telexistence)*, pages 1–7, Adelaide, Australia, December 2010. URL `https://hal.science/hal-00539074`. 23

[132] Zhiyuan Zhang, Xueqing Li, Lei Wang, and Peng Ma. Gpu-based collision detection between deformable objects. In *2008 5th International Conference on Visual Information Engineering (VIE 2008)*, pages 560–565, 2008. doi: 10.1049/cp:20080377. 23

[133] Min Tang, Huamin Wang, Le Tang, Ruofeng Tong, and Dinesh Manocha. CAMA: Contact-aware matrix assembly with unified collision handling for GPU-based cloth simulation. *Computer Graphics Forum (Proceedings of Eurographics 2016)*, 35(2):511–521, 2016.

[134] François Lehericey, Valérie Gouranton, and Bruno Arnaldi. Gpu ray-traced collision detection for cloth simulation. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, VRST '15, page 47–50, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450339902. doi: 10.1145/2821592.2821615. URL `https://doi.org/10.1145/2821592.2821615`. 23

[135] Yuxin Yang, Xiaoming Chen, and Yinhe Han. Dadu-CD: Fast and Efficient Processing-in-Memory Accelerator for Collision Detection. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. ISBN 978-1-72811-085-1. doi: 10.1109/DAC18072.2020.9218709. URL `https://ieeexplore.ieee.org/document/9218709/`. 24

[136] J. Chase Kew and Brian Ichter and Maryam Bandari and Tsang-Wei Edward Lee and Aleksandra Faust. Neural Collision Clearance Estimator for Batched Motion Planning, 2019. 24, 36, 37

[137] Nikhil Das, Naman Gupta, and Michael Yip. Fastron: An online learning-based model and active learning strategy for proxy collision detection. In *Conference on Robot Learning*, pages 496–504. PMLR, 2017. 24

[138] Yuheng Zhi, Nikhil Das, and Michael Yip. DiffCo: Auto-Differentiable Proxy Collision Detection with Multi-class Labels for Safety-Aware Trajectory Optimization, 2021. URL `http://arxiv.org/abs/2102.07413`. 24

[139] Yeseung Kim, Jinwoo Kim, and Daehyung Park. Graphdistnet: A graph-based collision-distance estimator for gradient-based trajectory optimization. *IEEE Robotics and Automation Letters*, 7(4):11118–11125, 2022. doi: 10.1109/LRA.2022.3196956. 24

[140] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore, 2022. 24

[141] Karol Cieślik and Marian J. Łopatka. Research on speed and acceleration of hand movements as command signals for anthropomorphic manipulators as a master-slave system. *Applied Sciences*, 12(8), 2022. ISSN 2076-3417. doi: 10.3390/app12083863. URL https://www.mdpi.com/2076-3417/12/8/3863. 25

[142] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17 (7):760–772, 1998. doi: 10.1177/027836499801700706. URL https://doi.org/10.1177/027836499801700706. 25

[143] T. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 388–393 vol.1, 2003. doi: 10.1109/IROS.2003.1250659. 25

[144] Satoshi Hoshino and Tomoki Yoshikawa. Motion planning of mobile robots for occluded obstacles. *Journal of Robotics and Mechatronics*, 30(3):485–492, 2018. doi: 10.20965/jrm.2018.p0485. 25

[145] Ran Zhao and Daniel Sidobre. Dynamic obstacle avoidance using online trajectory time-scaling and local replanning. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 02, pages 414–421, 2015. 25

[146] Federico Vesentini and Riccardo Muradore. Velocity obstacle-based trajectory planner for two-link planar manipulators. In *2021 European Control Conference (ECC)*, pages 690–695, 2021. doi: 10.23919/ECC54610.2021.9655184. 25

[147] Federico Vicentini, Matteo Giussani, and Lorenzo Molinari Tosatti. Trajectory-dependent safe distances in human-robot interaction. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4, 2014. doi: 10.1109/ETFA.2014.7005316. 26

[148] Paul Glogowski, Kai Lemmerz, Alfred Hypki, and Bernd Kuhlenkotter. Extended Calculation of the Dynamic Separation Distance for Robot Speed Adaption in the Human-Robot Interaction. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 205–212. IEEE, 2019. ISBN 978-1-72812-467-4. doi: 10.1109/ICAR46387.2019.8981635. URL https://ieeexplore.ieee.org/document/8981635/. 26

[149] Veo Robotics Inc. Veo FreeMove®, 2023. URL https://web.archive.org/web/20230519063645/https://www.veobot.com/freemove. 26

[150] Jia Pan, Ioan A. Sucan, Sachin Chitta, and Dinesh Manocha. Real-time collision detection and distance computation on point cloud sensor data. In *2013 IEEE International Conference on Robotics and Automation*, pages 3593–3599. IEEE, 2013. ISBN 978-1-4673-5643-5 978-1-4673-5641-1. doi: 10.

1109/ICRA.2013.6631081. URL `http://ieeexplore.ieee.org/document/6631081/`. 26

[151] Max Kaluschke, Uwe Zimmermann, Marinus Danzer, Gabriel Zachmann, and Rene Weller. Massively-Parallel Proximity Queries for Point Clouds. In Jan Bender, Christian Duriez, Fabrice Jaillet, and Gabriel Zachmann, editors, *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association, 2014. ISBN 978-3-905674-71-2. doi: 10.2312/vriphys.20141220. 26

[152] Mohammad Safeea, Pedro Neto, and Richard Bearee. Efficient calculation of minimum distance between capsules and its use in robotics. *IEEE Access*, 7: 5368–5373, 2019. doi: 10.1109/ACCESS.2018.2889311. 27

[153] Sezgin Secil and Metin Ozkan. Minimum distance calculation using skeletal tracking for safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*, 73:102253, 2022. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2021.102253. URL `https://www.sciencedirect.com/science/article/pii/S0736584521001332`. 27

[154] Oleynikova, Helen, Millane, Alexander, Taylor, Zachary, Galceran, Enric, Nieto, Juan, and Siegwart, Roland. Signed distance fields: A natural representation for both mapping and planning. ETH Zurich, 2016. doi: 10.3929/ETHZ-A-010820134. URL `http://hdl.handle.net/20.500.11850/128029`. 27

[155] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016. ISBN 978-0-9923747-2-3. doi: 10.15607/RSS.2016.XII.001. URL `http://www.roboticsproceedings.org/rss12/p01.pdf`. 27

[156] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018. doi: 10.1177/0278364918790369. URL `https://doi.org/10.1177/0278364918790369`. 27

[157] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. doi: 10.1109/ISMAR.2011.6092378. 27

[158] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[159] Luxin Han, Fei Gao, Boyu Zhou, and Shaojie Shen. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4423–4430, 2019. doi: 10.1109/IROS40897.2019.8968199. 27

[160] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022. 27

[161] Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. Simultaneous scene reconstruction and whole-body motion planning for safe operation in dynamic environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3710–3717, 2021. doi: 10.1109/IROS51168.2021.9636860. 27

[162] Christian Juelg, Andreas Hermann, Arne Roennau, and Rüdiger Dillmann. Fast online collision avoidance for mobile service robots through potential fields on 3d environment data processed on gpus. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 921–928, 2017. doi: 10.1109/ROBIO.2017.8324535. 27

[163] Andreas Hermann, Florian Drews, Joerg Bauer, Sebastian Klemm, Arne Roennau, and Ruediger Dillmann. Unified GPU voxel collision detection for mobile manipulation planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4154–4160. IEEE, 2014. ISBN 978-1-4799-6934-0 978-1-4799-6931-9. doi: 10.1109/IROS.2014.6943148. URL `http://ieeexplore.ieee.org/document/6943148/`. 27

[164] Thanh-Tung Cao, Ke Tang, Anis Mohamed, and Tiow-Seng Tan. Parallel banding algorithm to compute exact distance transform with the gpu. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 83–90, 2010. 27

[165] Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Regularized deep signed distance fields for reactive motion generation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6673–6680, 2022. doi: 10.1109/IROS47612.2022.9981456. 28

[166] Kris Hauser. Semi-infinite programming for trajectory optimization with non-convex obstacles. *The International Journal of Robotics Research*, 40 (10-11):1106–1122, 2021. doi: 10.1177/0278364920983353. URL `https://doi.org/10.1177/0278364920983353`. 28

[167] Shan Xu, Gaofeng Li, and Jingtai Liu. Obstacle Avoidance for Manipulator with Arbitrary Arm Shape Using Signed Distance Function. In *2018 IEEE*

International Conference on Robotics and Biomimetics (ROBIO), pages 343–348. IEEE, 2018. ISBN 978-1-72810-377-8. doi: 10.1109/ROBIO.2018.8665083. URL `https://ieeexplore.ieee.org/document/8665083/`. 28

[168] Jingru Luo and Kris Hauser. An empirical study of optimal motion planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1761–1768. IEEE, 2014. 31, 32

[169] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, 2020. 36

[170] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2015. 37

[171] Rosen Diankov. ParabolicSmoother - rplanners: OpenRAVE Documentation, 2021. URL `http://openrave.org/docs/latest_stable/interface_types/planner/parabolicsmoother/`. Last accessed 15 September 2021. 39, 40

[172] Puttichai Lertkultanon and Quang-Cuong Pham. Time-optimal parabolic interpolation with velocity, acceleration, and minimum-switch-time constraints. *Advanced Robotics*, 30(17-18):1095–1110, 2016. doi: 10.1080/01691864.2016.1204247. URL `https://doi.org/10.1080/01691864.2016.1204247`. 39

[173] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL `http://www.programmingvision.com/rosen_diankov_thesis.pdf`. 39, 63, 74

[174] Gary Miller, Yanzhe Yang, and Yao Liu. Lecture 16: Linear Programming in Fixed Dimensions. 2016. 54

[175] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 56

[176] Tianyi David Han and Tarek S. Abdelrahman. Reducing branch divergence in gpu programs. In *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-4, pages 1–8, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305693. doi: 10.1145/1964179.1964184. URL `https://doi.org/10.1145/1964179.1964184`. 56

[177] Hung Pham. Hungpham2511/toppra: Robotic motion planning library, 2018. URL `https://github.com/hungpham2511/toppra`. 63

[178] Shohei Fujii and Quang-Cuong Pham. Time-Optimal Path Tracking with ISO Safety Guarantees. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5926–5933, October 2023. doi: 10.1109/IROS55552.2023.10342287. URL https://ieeexplore.ieee.org/document/10342287. 67, 75

[179] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf. 70

[180] Dominik Ernst, Georg Hager, Jonas Thies, and Gerhard Wellein. Performance engineering for real and complex tall & skinny matrix multiplication kernels on gpus. *The International Journal of High Performance Computing Applications*, 35(1):5–19, 2021. doi: 10.1177/1094342020965661. URL https://doi.org/10.1177/1094342020965661. 72, 73

[181] Ozgur S. Oguz, Zhehua Zhou, Stefan Glasauer, and Dirk Wollherr. An inverse optimal control approach to explain human arm reaching control based on multiple internal models. *Scientific Reports*, 8(1), April 2018. doi: 10.1038/s41598-018-23792-7. URL https://doi.org/10.1038/s41598-018-23792-7. 85

[182] Rafael A. Rojas, Manuel A. Ruiz Garcia, Erich Wehrle, and Renato Vidoni. A variational approach to minimum-jerk trajectories for psychological safety in collaborative assembly stations. *IEEE Robotics and Automation Letters*, 4(2):823–829, 2019. doi: 10.1109/LRA.2019.2893018. 85

[183] Hung Pham and Quang-Cuong Pham. On the structure of the time-optimal path parameterization problem with third-order constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 679–686, 2017. doi: 10.1109/ICRA.2017.7989084. 85

[184] A. T. Welford, J. M. T. Brebner, N. Kirby, D. McNicol, T. Nettelbeck, G. A. Smith, G. W. Stewart, and D. Vickers. *Reaction times*. Academic Press, New York, 1980. 85

[185] Francisco Suárez-Ruiz, Xian Zhou, and Quang-Cuong Pham. Can robots assemble an ikea chair? *Science Robotics*, 3(17):eaat6385, 2018. doi: 10.1126/scirobotics.aat6385. URL https://www.science.org/doi/abs/10.1126/scirobotics.aat6385. 87

[186] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. I-planner: Intention-aware motion planning using learning-based human motion prediction. *The International Journal of Robotics Research*, 38(1):23–39, 2019. doi: 10.1177/0278364918812981. URL https://doi.org/10.1177/0278364918812981. 88

[187] Ziwen Wan, Junjie Shen, Jalen Chuang, Xin Xia, Joshua Garcia, Jiaqi Ma, and Qi Alfred Chen. Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks. In *Network and Distributed System Security (NDSS) Symposium, 2022*, April 2022. 88