Time-optimal motion planning and control

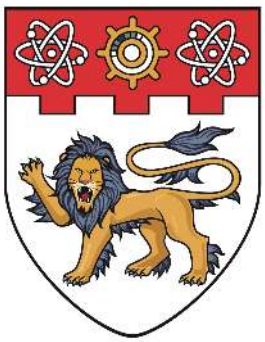
Pham, Tien Hung

2019

Pham, T. H. (2019). Time-optimal motion planning and control. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/83546

https://doi.org/10.32657/10220/49772



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

TIME-OPTIMAL MOTION PLANNING AND CONTROL

PHAM TIEN HUNG

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

2019

TIME-OPTIMAL MOTION PLANNING AND CONTROL

PHAM TIEN HUNG

School of Mechanical and Aerospace Engineering

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

2019

PHAM TIEN HUNG
Time-optimal Motion Planning and Control

Nanyang Technological University
Control Robotics Intelligence Group (CRI)
School of Mechanical and Aerospace Engineering
Robotics Research Centre
50 Nanyang Avenue
N3-01a-01
Singapore 639798

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 Feb 2019

....

Date Pham Tien Hung

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 Feb 2019

.....
Date

.....
Quang-Cuong Pham

Authorship Attribution Statement

This thesis contains material from 6 papers published in the following peer-reviewed journals and conferences where I was the first and/or corresponding author.

Chapter 3 is published as Hung Pham and Quang-Cuong Pham. A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. IEEE Transactions on Robotics, Jul 2018. ISSN 15523098. DOI: 10.1109/TRO.2018.2819195.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts and performed all experiments.

Chapter 4 is published as Hung Pham and Quang-Cuong Pham. Critically fast pick-and-place with suction cups. Accepted at 2019 IEEE International Conference on Robotics and Automation (ICRA), 2019.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts and performed all experiments.

Chapter 5 is published as Hung Pham and Quang-Cuong Pham. On the Structure of the Time-Optimal Path Parameterization Problem with Third-Order Constraints. In Allison M. Okamura, editor, 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017. IEEE. ISBN 9781509046324.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts and performed all experiments.

Chapter 6 is published as Hung Pham and Quang-Cuong Pham. Time-Optimal Path Tracking via Reachability Analysis. In 2018 IEEE International Conference on Robotics and Automation, 2018. ISBN 9781538630808.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts and performed all experiments.

The first paper in Appendix A is published as Hung Pham and Quang-Cuong Pham. Robotic Manipulation of a Rotating Chain. IEEE Transactions on Robotics, Apr 2018.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts and performed all experiments.

The second paper in Appendix A is published as Hung Pham, Lim Jian Hui and Quang-Cuong Pham. Robotic 3D-printing for building and construction. In Proceedings of the 2nd International Conference on Progress in Additive Manufacturing (Pro-AM 2016), 2016.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction and edited the manuscript drafts.
- I prepared the manuscript drafts and performed all the simulation.
- Jian Hui Lim wrote a section in the manuscript drafts.

21 Feb 2019

....

Date Pham Tien Hung

Acknowledgements

I would like to thank my supervisor Quang-Cuong Pham for introducing me to the wonderful world of scientific research; for teaching me to look for the big picture and to ask the right questions; and most of all for patiently guiding me through this tiring, laborious and frustrating but simultaneously rewarding and exciting journey.

I am eternally grateful to my parents for teaching me the value of hard work without which this thesis would never materialize, to my sister Ngoc who I own great debt and to my partner Tam for her unwavering trust and support when I need them the most.

Last but not least, I would like to thank my friends at CRI group–Huy Nguyen, Jian Hui Lim, Xu Zhang, Francisco Suárez-Ruiz, Puttichai Lertkultanon, Xuan Hien Bui, Nicholas Adrian and many others–for the stimulating discussions and all the fun during our lab trips. You guys have made this journey infinitely more memorable.

Abstract

The recent years have seen rapid growth in the adoption of robotics technologies. This welcoming development has led to increasingly complex applications with stringent requirements, motivating research on time-optimal motion planning and control for robots.

This thesis presents developments that extend the state-of-the-art in time-optimal *motion planning* and *control* for robots. I first revisit a classical problem in the robotic literature–computing the Time-Optimal Path Parameterization along a specified path–which was posed more than 30 years ago by Bobrow et al. [16]. The presented new approach to the problem, as suggested by experimental evaluations, outperforms existing solutions in both computational complexity and robustness. Next, I discuss an experimental case study of an industrial task: planning *critically fast* motions for robots transporting objects with suction cups. Experimental results suggest that by appropriately modelling of "suction cup constraints", one can control an industrial robot at high speed (near the robot hardware speed limit) and still achieve 100% transport success rate. Finally, I discuss and present solutions to two issues that are commonly associated with the use of Time-Optimal Path Parameterizations: (i) the existence of switching points with infinite joint jerk and (ii) the poor regulation of tracking error.

Contents

Chapter 1.

Introduction

1.1. Motivation

The recent years have seen a tremendous growth of robotic adoption in different industries. The International Federation of Robotics (IFR) reported an increase of 30% in the total number of industrial robot sales in 2017, continuing an upward trend since 2010 [60]. The main drivers of this growth are the automotive industry, the electrical/electronics industries and the metal industry [60]. The impacts of robotics, however, extend well beyond the traditionally known factories setting, penetrating other areas including logistics [19]. To name a few, Automated Guidance Vehicles [81, 126] (AGV) have been used in large warehouses [129] for autonomously moving packages in between areas or on the streets for last-mile delivery [21]; in the forthcoming years, automated bin-picking with industrial robots is expected to bring the operational throughput of warehouses [30, 85] to the next level.

Not limiting to the industrial settings that it originated from, robotics has also been recognized and hence, widely applied in other non-industrial contexts as well. For instance, looking at surveillance and rescue, legged robots such as quadrupedal robots [59, 87] and humanoids [45, 57] are currently used to provide services on difficult terrains which are inaccessible to traditional wheeled alternatives (e.g. stairs, rocks or ragged surfaces). Autonomous driving [4] is another exciting non-industrial field that has enjoyed advanced technologies that have been built upon many years of robotic research.

However, to deploy these novel robotic applications in practice, the traditional approach to robotic development does not suffice. Indeed, comparing to the traditional applications, these modern applications have become much more complex: There are challenges that render the conventional approach inadequate. Before going into details

about these challenges, let us first take a look at the conventional approach.

Traditional industrial robotic applications are relatively simple: a fixed surrounding environment with precisely known geometries in an isolated cell. Examples include pick-and-place/material handling from one fixed position to another, welding along fixed paths and assembly parts with precise position measurements. In this class of applications, engineers "teach" robots fixed motion trajectories by specifying waypoints [77], which are then executed repetitively by simple conditionals. Having the robots repeating a set of pre-defined motions is not only sufficient for these traditional applications but also advantageous, as it leads to a higher level of predictability and a lower failure rate. Nonlinear optimization (e.g. simulated annealing, genetic algorithms) can be performed offline to enhance the motion trajectories [49] with respect to certain optimization objectives and constraints.

In the modern applications, however, one can no longer assume that the external environments surrounding the robots *will remain fixed*. Indeed, an example is the collaborative human-robot work cells [27]; here the operator is an extremely unpredictable component of the environment. Similarly, environments such as warehouses (AGVs, pick-and-place robots) and streets (autonomous vehicles, delivery mobile robots) are both unstructured, varying and sometimes uncertain. Furthermore, many applications have non-trivial task dynamics. Consider bin picking, a challenge in this task is to respect *the frictional constraints* between the end-effector and the object in order to successfully and robustly transport objects. With this understanding, it is clear that the traditional approach of teaching robots manually is rather impractical for the modern applications. Additionally, in many applications, the initial and target poses of the robots are only known during execution, further emphasizing the impracticality of the manual teaching approach. At this point of writing, the alternative of computational motion planning is also not a straightforward solution for modern applications. The main reason is the computational challenges associated with handling complex and non-linear robot and task dynamics in motion planning. On top of that, the stringent requirements of low computational complexity and high robustness in industrial applications are other hurdles that need to be addressed.

The above elaboration is far from being an exhaustive list of the challenges. Indeed, deploying robotic systems that are sufficiently efficient and robust for modern applications requires breakthroughs on multiple fronts of robotics: *motion planning, control*, manipulation and computer vision. *Motion planning and control* are the main focuses of this thesis. In the following, I discuss how the challenges of modern robotic applications affect these two sub-fields in more detail:

1. *Achieve high throughput while satisfying hard constraints*

   In many industrial tasks, the most important metric is the total throughput, e.g.

the number of successfully picked items in bin-picking or miles travelled in autonomous vehicle. This is because this metric directly correlates with the return on investment of the robotic system. Hence, moving the robots in a time-optimal manner, which subsequently maximizes the total throughput, is desirable.

At the same time, robotic systems need to satisfy multiple *hard* constraints– constraints that lead to system failure or poor performance if violated. An example is the motors' joint current limits. If the robot executes a motion while carrying a heavy payload, some motor currents might violate the respective limits which, in many industrial robots, causes the control system to shutdown to prevent further damage. Another example is the grasp stability constraint: the contact force between the suction cup/ gripper with the object being transported must satisfies frictional limits throughout the movements. A robot moving too fast could violate this constraints and thus risks failing the grasp.

The key challenge is to plan the fastest or time-optimal motion that still guarantees constraint satisfaction. Another associated challenge is to track these time-optimal motions, which is notoriously known to be difficult.

2. *Handle complex robot and task dynamics*

   Robot dynamics is nonlinear, complex and generally difficult to handle computationally. Most robots consist of multiple links, which are rigid bodies, joint together via rotational or transnational joints [116]. The dynamics of a robot with n degree-of-freedom (joints) [79] is:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau},$$

   where $\mathbf{M}, \mathbf{C}, \mathbf{g}$ are matrix, third-order tensor and vector functions of the robot physical parameters and joint position; $\mathbf{q}, \boldsymbol{\tau}$ are respectively the joint positions and the joint torque. The above dynamical equation is multi-dimensional, nonlinear and usually is unavailable in closed-form.

   In addition, to guarantee successful execution in many tasks such as bin-picking and legged robot locomotion, robot motions have to satisfy additional task-specific constraints such as frictional constraints. In bin picking, the contact between the suction cup and the object might fail if the maximal frictional contact force is insufficient to hold the object [98]. Similarly, in legged robot locomotion, if frictional forces between the robot and the environment cannot support the robot in its movement, slippage or lost of contact will occur [26], leading to failure. To generate truly dynamic motions for these complex tasks, the motion planner needs to account for the task constraints simultaneously with the robot's

rigid body dynamics.

3. *Achieve high operational performance: low computational complexity and high robustness*

   Computational complexity and robustness are in some sense new challenges in robotics. In a traditional robotic system, the robot mostly repeats pre-defined motions; there is very little online computation. In a modern robotic system, on the other hand, motion planning must be performed online due to the complexity of the applications. However, numerical algorithms can be slow and prone to failure. Both are detrimental to the overall performance of a robotic system, and therefore require special attention.

   Most robotic tasks are cyclic: a control/task cycle consists of computational time and execution time. The benefit of using time-optimal motions can be outweighted by the high computational cost. For instance, if the an optimal motion that lasts 1 seconds is computed in 5 seconds, it is probably better to choose a non-optimal motion that lasts 3 seconds with only 0.5 seconds of computational time. Therefore, computational cost is a critical factor that one has to account for in designing time-optimal motion planning algorithms.

   In addition, a motion planning instance could fail, either because the task is physically impossible, or because of numerical issues. Both cases are undesirable, as they require manual intervention from operators or additional error recovering procedures. Ensuring that there exists at least a physically possible motion is the easier job, since industrial robots are capable of very rapid and smooth movements. On the other hand, it is much harder to ensure that a numerical algorithm always work correctly for ill-conditioned inputs.

The three aforementioned difficulties are long standing problems in robotics. In this thesis, I will present developments that seek to address them. The first contribution is an efficient and robust algorithm for computing the time-optimal path parameterization for a wide class of robots and tasks. The second contribution is an experimental case study of the challenging task of transporting objects with suction cup *critically fast*. As the third and fourth contributions, I will present analyses of two well-known issues associated with time-optimal motions in practice: infinite joint jerk at the switching points of time-optimal trajectories and poor tracking performance of time-optimal trajectories. The next section discusses each contribution in more detail.

1.2. Contributions

1.2.1. Time-Optimal Path Parameterization based on Reachability Analysis

Time-Optimal Path Parameterization (TOPP)–the problem of computing the time-optimal parameterization of a geometric path–is a classical problem in the robotic literature [16]. It is a key step in the Path-Velocity Decomposition motion planning principle [61] that is widely used in industrial robotics. There are two main families of methods to solve TOPP: Numerical Integration [16, 119, 99] and Convex Optimization [125]. Numerical Integration-based methods are fast but are difficult to implement and suffer from robustness issues. Whereas Convex Optimization-based methods are relatively more robust but significantly slower.

A contribution of this thesis is a new approach to TOPP based on Reachability Analysis [10]. The proposed algorithm achieves the best of both worlds: it is faster than Numerical Integration-based methods and as robust as Convex Optimization-based ones, as confirmed by extensive numerical evaluations. Moreover, the proposed approach offers unique additional benefits: Admissible Velocity Propagation [101] and robustness to parametric uncertainty as demonstrated in Chapter 4 and [97] can be derived in a simpler and more natural way.

1.2.2. Critically fast pick-and-place with suction cups

Pick-and-place with suction cups is a typical application of robotics, in both factories and logistic settings [30, 85]. Here, we aim to plan "critically fast" motions, which are motions that are the fastest possible for transporting an object while maintaining the object-suction cup contact throughout the motion.

The main difficulties of planning critically fast motions are: (i) accounting for the contact stability constraints that arise from maintaining the object-suction cup contact while (ii) achieving a low computational cost and a high level of robustness. To address these difficulties, the following developments were made: (a) a model for object-suction cup contacts, (b) a procedure to identify contact stability constraints, and (c) a procedure to parameterize time-optimally arbitrary geometric paths while accounting for contact stability constraints. A series of experimental evaluations demonstrates that an industrial robot, by accounting for suction cup-object contact stability constraints, can achieve 100% success rate; when contact stability constraints are not accounted for, the success rate plunges to merely 20%. The full pipeline is released as open-source for the robotics community.

1.2.3. Structure of the Time-Optimal Path Parameterization Problem with Third-Order Constraints

Time optimal motions contain switching points [84]: points at which one or multiple joints switch from maximally accelerating to maximally decelerating or vice versa. Executing time-optimal motions on industrial robots, therefore, damages the motors and causes vibrations, especially for large robotic arms [86]. Here, we approach this problem by computing time-optimal parameterizations that also account for joint jerk or torque-rate constraints.

This thesis presents an investigation on the structure of the Time-Optimal Path Parameterization problem with third-order constraints and shows that there are two major difficulties: (i) how to smoothly connect optimal profiles, and (ii) how to address singularities, which stop profile integration prematurely. Resulting from this investigation is a development of a new algorithm which addresses these two difficulties and thereby constitutes an important milestone towards an efficient computational solution to the Time-Optimal Path Parameterization with third-order constraints.

1.2.4. Time-Optimal Path Tracking via Reachability Analysis

Another issue commonly associated with time-optimal motions is the difficulty in regulating tracking error during execution. The problem lies in the bang-bang property of time-optimal motions [84]: at any moment, there is at least one saturated constraint. While this is normally inconsequential, when the saturated constraint is a joint torque limit, the robot controller is incapable of producing additional torque to regulate tracking error, hence leading to poor tracking performance.

The fourth contribution is a new solution for the Time-Optimal Path Tracking problem: design a path tracking controller (also known as path following controller [1]) that traverses the path time-optimally. Existing solutions are either unable to guarantee exponential convergence of tracking error (Online Scaling controllers [33]) or overly conservative and require hand-designed terminal sets (Model Predictive Control [40]). The proposed solution can guarantee exponential convergence of tracking error and at the same time achieve near time-optimal trajectory duration.

1.3. Outline of the Thesis

Related literature are reviewed in Chapter 2. Chapters 3, 4, 5, and 6 present the main contributions. Conclusion and perspective on future research are given in Chapter 7. A list of publications that results from this thesis can be found in Appendix A.

Chapter 2.

Literature Review

In this chapter, I review the literature related to the time-optimal path parameterization problem and discuss the position of the contributions of this thesis to this broader literature.

2.1. Path parameterization overview

In many industrial robotic systems, motion planning follows the Path-Velocity Decomposition principle, which was introduced by Kant and Zucker [61]. In this principle, one computes a joint trajectory for a robot with n degrees-of-freedom (dof) in two consecutive stages: *geometric path planning* and *path parameterization*. In geometric path planning, a path planner (e.g. RRT [69], CHOMP [132]) plans a collision-free geometric path, which is a continuous function of a scalar *path position* variable s:

$$\mathbf{q}(s) \in \mathbb{R}^n, s \in [0, 1].$$

Here $\mathbf{q}(s)$ maps the path position s to the joint configuration of the robot. In path parameterization, a second planner retimes the geometric path by computing a time parameterization $s(t) \in [0, 1], t \in [0, T_{end}]$. This is commonly known as the path parameterization problem [16]. The composition of the geometric path $\mathbf{q}(s)$ and the time parameterization $s(t)$ is the final trajectory

$$\mathbf{q}(t) := \mathbf{q}(s(t)) \in \mathbb{R}^n, t \in [0, T_{end}].$$

Robot and task constraints are handled in the path parameterization stage.

In the robotic literature, researchers have investigated path parameterization prob-

lems with various kinds of constraints. These constraints come from the need to plan motions for increasingly complex robotic systems. In the pioneering papers, Bobrow et al. [16], Shin and McKay [113], Slotine and Yang [119] considered joint torque and acceleration limits for fully-actuated robotic manipulators. Subsequently, Zlajpah [131] and Kunz and Stilman [66] considered path parameterization problems with additionally joint velocity constraint and proposed solutions. More recently, to plan motion for humanoids and redundant manipulators interacting with the environment, Pham and Stasse [100], Bobrow et al. [17], Larochelle et al. [68] investigated the joint torque bound constraint for redundantly actuated robots. Subsequently, Hauser [55], Caron et al. [26] considered constraints resulting from frictional contact such as in humanoid contacts for balancing, fast non-prehensile object transportation [78, 31] or pick-and-place with suction cup [98]. To obtain smoother motions, Debrouwere et al. [34], Pham and Pham [95], Kaserer et al. [62] investigated jerk or torque rate constraints. Recently, Pham [99] introduced a constraint class that consists of many constraints frequently occurred in practice and can be handled efficiently. This class was later extended in Chapter 3.

Researchers have also formulated path parameterization problems with different optimizing objectives. In his seminal work, Bobrow et al. [16] introduced the minimum-time criterion for computing path parameterizations. The minimum-time criterion remains the most commonly studied objective in the literature [113, 131, 99, 54, 96]. One reason is that time-optimality corresponds directly to the maximal production output. Subsequently, Shiller [109] introduced the time-energy objective and showed that by trading execution speed, smoother trajectories can be achieved. With the same goal of improving smoothness, Kyriakopoulos and Saridis [67], Bianco [13], Piazzi and Visioli [102] considered minimum-jerk (the third derivative of joint position) objective. However, minimal-jerk trajectories can be overly conservative and the high level of smoothness might not bring significant improvement to the control [1]. General cost functions were also considered. By discretizing the state-space and using Dynamic Programming, Shin and McKay [114] considered very general cost functions. More recently, Verscheure et al. [125] reformulates the path parameterization problem as a convex optimization problem, which naturally allow the consideration of generic convex objectives.

In subsequent sections of the chapter, we review different aspects of Time-Optimal Path Parameterization (TOPP)–the problem of computing the fastest possible time parameterization for a given geometric path without violating constraints. In particular, we first discuss different approaches to solving TOPP, including the approach proposed in this thesis (Section 2.2). Next, we discuss applications of TOPP, both as a proce-

[1] Notice that this is different from generating trajectories with *bounded* jerk

dure to plan motion directly and as a sub-routine supporting other motion planning algorithms (Section 2.3). Literature on *smooth* path parameterization by including third-order constraints, e.g. joint jerk and torque-rate bounds, is in Section 2.4. Lastly, we discuss issues related to *tracking* time-optimally parameterized geometric paths (Section 2.5).

2.2. Time-optimal path parameterization (TOPP): Algorithms

In the literature, there are three main approaches to TOPP that guarantee finding the optimal solution: Numerical Integration (NI), Dynamics Programming (DP) and Convex Optimization (CO). We discuss each approach in the respective order. We also discuss the non-convex optimization approach, which does not guarantee finding the optimal solution.

2.2.1. Numerical integration

Bobrow et al. [16] proved that the optimal time-parameterization consists of alternatively maximally accelerating and decelerating segments and suggested the first NI-based algorithm to solve TOPP. Around the same period, Pfeiffer and Johanni [94], Shin and McKay [113] developed other NI-based algorithms that follow the same basic principle. All of these initial NI-based algorithms involve expensive computations for finding *switching points*–which are points at which the velocity profile changes from maximally decelerating to maximally accelerating [16]. A development was made by Slotine and Yang [119], who classified the switch points into *tangent, singular and discontinuous switch points* and introduced procedures for finding them. This development leads to significant improvement in computational cost. Later on, Zlajpah [131] introduced *trap regions* to handle joint velocity constraints in NI-based algorithms.

These initial NI-based algorithms, however, were prone to failure in the presence of *singular switch points*, also known as *zero-inertia points* [119]. Only until recently, Kunz and Stilman [66] developed an NI-based algorithm that can handle singular switch points for robots subject to joint velocity and acceleration constraints in a realiable manner. Subsequently Pham [99] proposed a more general approach that can account for singular switch points resulting from "Second-Order Constraints", which include joint torque bound, Zero Moment Point constraint, frictional constraint in addition to joint acceleration bounds.

The state-of-art NI-based algorithm proposed by Pham [99] is fast (terminate in

milliseconds) and relatively robust (failure rate in a randomized test with 1000 trajectories is 0.1% [99]). Yet, it has two weaknesses. First, computing the switch points is difficult and numerically sensitive. While the proposed solutions [99, 66] are effective to a certain extent, this remains a major implementation difficulty as well as the main cause of failure. Second, finding the switch points and computing the Maximum Velocity Limit curve become computationally expensive for complex constraints with many inequalities [96, 98].

2.2.2. Dynamic Programming

The Dynamic Programming algorithm, discovered by Bellman [5], is among the most important tools in computer science [29], operational research [9], optimal control [121] and reinforcement learning [123]. Shin and McKay [114] were the first researchers who used Dynamic Programming to solve TOPP. They discretized uniformly the path interval $[0, s_{end}]$ and the path velocity interval into grids with finite points and applied the DP algorithm to solve the discretized problem. This approach has two advantages: simple implementation and the ability to account for general cost functions and constraints, including third-order constraints. The main disadvantage, however, is the high computational cost required to solve finely discretized instances, which are compulsory for generating high quality trajectories.

2.2.3. Convex optimization

Convex optimization [20] is a branch of mathematical programming dealing with optimization problems that have convex constraints and objectives. These problems are of special interest in many branches of engineering because "there are effective methods for solving them" [20]. Another important advantage of convex problems is that any local optimal solution is also globally optimal [20].

In 2008, Verscheure et al. [125] showed that TOPP can be formulated as a second-order cone (convex) program. With the availability and maturity of second-order cone solvers, this approach can reliably and robustly solve TOPP. The main disadvantage is computational cost, which can be order of magnitude greater than NI-based approaches. Subsequent developments by Hauser [54], Lipp and Boyd [75] further reduce computational cost. Hauser [54] proposed an iterative Linear Programming algorithm [122], which was combined with a pruning procedure to significant reduce computational cost. Recently, Nagy and Vajk [88] showed that using a special discretization scheme, the globally optimal solution is found by solving a single Linear Program.

The best computational complexity achieved by a CO-based approach at the current

point of writing is the same as solving a single Linear Program [88], whose decision variables are the squared path velocities at each grid point. This computational complexity is $O(mN^3)$ [96] where N is the number of grid points and m is number of scalar constraints. Comparing to the state-of-art NI-based algorithm [99], which has computational complexity $O(m^2N)$, this is still order of magnitude slower because the number of gridpoints N is often higher than the number of constraints m. This makes CO-based algorithms inappropriate for online motion planning or as subroutine to kinodynamic motion planners [101]. This thesis presents a new approach that further lowers the computational complexity $O(m^2N)$ of [99].

2.2.4. Non-convex optimization

For arbitrary dynamic constraints and optimization objective, one can always formulates TOPP as a non-convex nonlinear constrained optimization problem:

$$\begin{aligned} \min \quad & f(s(t)) \\ & s(t) \in \mathcal{S}. \end{aligned} \tag{2.1}$$

Here, $\mathcal{S}$ is the set of feasible velocity profiles $s(t)$ that satisfies all system constraints. In principle, this optimization problem can be solved with nonlinear programming [8]. Several researchers have taken this approach to TOPP: Constantinescu and Croft [28], Gasparetto and Zanotto [47, 47] proposed to represent the time parameterization as a spline in the velocity phase plane and use the spline coefficients as the optimization variables. The optimization objective and constraints are translated to objective and constraints on the spline coefficients. A nonlinear constrained optimizer is then used to solve the resulting optimization problem.

This approach is general, but has two principal disadvantages. First, there is no guarantee that the solution is a globally optimal one. This is different from the CO-based approaches that we discussed in Section 2.2.3. As a result, the quality of solutions (which are locally optimial) depends heavily on the initial guesses, leading to inconsistency in motion quality. Second, non-convex optimization is time consuming and sometimes requires manual hyper parameter tuning to achieve good results.

2.3. Time-Optimal Path Parameterization in robotic applications

The TOPP problem has appeared in many applications of robotics and control engineering, ranging from humanoid locomotion [23], robotic pick-and-place [98] to ma-

glev positioning control system [90]. This section surveys the literature and discusses the relevant applications.

A fruitful approach to planning highly dynamic robot motions is to reformulate it as a TOPP instance. Bobrow et al. [16], McCarthy and Bobrow [84] planned motions for robots with limited joint torque transporting heavy payloads by formulating and solving TOPP. In more recent works, researchers proposed different TOPP formulations to solve the "waiter motion" problem [70, 43, 78, 31]. This problem consists in a robot, equipped with a flat plate as its end-effector, transports an object using only frictional and gravitational forces. Lertkultanon and Pham [70] included the Zero Moment Point [128] constraint in their TOPP formulation to ensure that the object does not fall. Additionally, Csorvási et al. [31], Debrouwere et al. [34] proposed to include both dry and viscous friction constraints to prevent slipping. Luo and Hauser [78] developed an iterative learning procedure to learn the coefficient of friction between the plate and the object in multiple trials. Following this line of work, in Chapter 4, I present a method to plan critically fast motions for transporting objects with suction cup by formulating and solving TOPP. Finally, it has also been demonstrated that one can plan highly dynamic motions for humanoids with frictional and balance constraints [54, 53, 100] by formulating and solving TOPP instances.

Multiple researchers have formulated TOPP instances in sub-routines of their high-level motion planning algorithms. Caron and Pham [23] proposed a Model-Predictive Control controller that solves a TOPP instance in every control cycle. Pham et al. [101] developed the Admissible Velocity Propagation (AVP) algorithm for kinodynamic motion planning [69], which grows a RRT tree [65] whose nodes are associated with reachable velocities from the initial state. AVP solves multiple TOPP instances to compute the interval of admissible velocities.

In control engineering, the fundamental idea underlying TOPP–decomposing a trajectory into the underlying geometric path and the time-parameterization–has also appeared as the *path tracking problem* [1, 90]. Tracking geometric paths is, in many cases, a more natural control objective than tracking trajectories [40]. Some representative applications are path tracking for autonomous vehicles [1], maglev positioning control system [90], underwater vehicles [39] and ships [36]. A discussion of the relationship between TOPP and path tracking is given in Section 2.5.

2.4. Time-Optimal Path Parameterization with third-order constraints

The methods surveyed in the Section 2.2 can handle TOPP instances with first-order constraints (e.g. velocity bounds, momentum bounds) and second-order constraints

(e.g. joint torque bound, acceleration bounds). However, problems with third-order constraints, are much less understood. We survey some of the attempts in the literature below.

Shin and McKay [114] proposed the first approach to TOPP with third-order constraints based on Dynamic Programming. However, it was noted in [114]: "To solve the optimization problem with jerk constraints using dynamic programming, a three-dimensional grid is required...". As a result, the computational cost significantly increases, presenting a significant obstacle to its adoption in practice. Recently, Kaserer et al. [62], Oberherber et al. [91] extended this approach. They proposed to use splines to interpolate in the two-dimensional velocity space, thereby avoiding discretizing the acceleration dimension. This approach reduces computational cost, but the final solution is not the true optimal solution.

Tarkiainen and Shiller [124] initiated the use of the Numerical Integration approach. In the seminal paper [124], they used Pontryagin's Maximum Principle to show that, similar to TOPP with only first-order and second-order constraints, the optimal trajectory is *necessarily* bang-bang. At any moment, at least one constraint saturates. This result is central to the Numerical Integration-based algorithm proposed by Tarkiainen. However, this algorithm has two limitations:

- It assumed that the optimal velocity profile has a maximum-minimum-maximum jerk structure. In generally, this is only true for the simplest cases.
- Path parameterization instances often contain multiple *singularities* that stop numerical integrations prematurely, hence, causing failure [94]. Tarkiainen's algorithm can not account for singularities.

In the robotic literature, there is surprisingly little research on these limitations. In 2009, Mattmüller and Gisler [82] proposed an approximate method to compute near-optimal profiles: they introduced "split points" artificially to "guide" the profiles away from singularities. While their method has the advantage of simplicity, it is sub-optimal and does not help understanding the structure of the true time-optimal solutions. Further, the points must be chosen manually, hence, requiring manual intervention for each new path. In Chapter 5, I present (i) an analysis of third-order singularities and a method to address them as well as (ii) an algorithm that can realize more general switching structures.

Dong et al. [38] developed an interesting solution to this problem. Computations are divided into two phases. In the first phase, using a Numerical Integration-based algorithm, a velocity profile that satisfies first-order and second-order constraints is computed. Necessarily this profile violates third-order constraints at certain points. In the second phase, the authors proposed to scan this velocity profile for "jerk-discontinuations"

and fix these points locally. An assumption of this algorithm is that the jerk-limited solution is similar to the one found in the first phase. However, this might not hold true in cases that third-order constraints are very restrictive.

The presence of third-order constraints also causes difficulties to optimization-based approaches: Verscheure et al. [125] showed that the resulting optimization problem is not convex. The advantages of convex optimization: global optimality, feasibility and robustness are, therefore, no longer guaranteed. Alternatively, Zhang et al. [130] approximated jerk constraint by linear functions, effectively convexified the problem. More recently, Singh and Krishna [118] proposed a representation of velocity profiles by a class of C^∞ functions. This representation ensures the continuities of higher-order derivatives such as jerk, snap, etc., while still allowing for efficient solutions via convex optimization. Nonlinear non-convex programming-based approaches were proposed by several researchers [28, 12, 76, 46]. In general, velocity profiles are approximated by piece-wise polynomials. Nonlinear programming is then used to find the locally optimal polynomial coefficients. Remark that since the optimization is non-convex, there is no guarantee of global optimality.

2.5. Time-optimal Path Tracking

Tracking time-optimally parameterized geometric paths on a real robot is difficult. An intuitive explanation of this issue is that if the robot is lagged behind, it can never "catch up" because the reference trajectory is time-optimal. More precisely, note that time-optimal trajectories are bang-bang [84]: At any moment, the feed-forward torque of at least one joint is at the joint's maximum or minimum limits. As a result, there is no available torque to correct tracking errors or to reject disturbances, resulting in poor tracking performance. A simple solution, therefore, is to consider a more conservative torque bounds during planning, "reserving" some torques for tracking during execution [64]. This approach has the advantage of simplicity, but the resulting motion could be conservative.

An alternative approach [33] is to modify the time-parameterization online in order to, for instance, slow down the reference trajectory to allow the robot to catch up. In fact, the reference time-parameterization is not necessarily time-optimal but can be a nominal time-parameterization [15]. Researchers have also considered the problem of purely tracking the geometric path [1, 40] without a pre-defined time-parameterization. All three problems can be classified as instances of the path tracking (also known as " Path-Following") problem [1], which in many cases is a more natural description of the control problem, compared to trajectory tracking or set-point tracking [40]. Several applications in control engineering demonstrate this point: Autonomous ve-

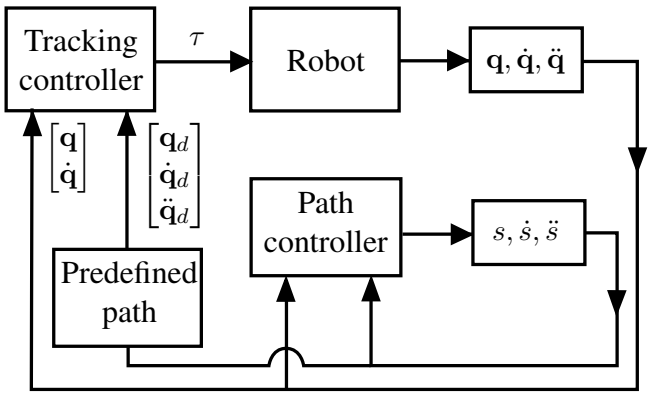


Figure 2.1.: Block diagram of an Online Scaling controller.

hicles [1], ships [36], autonomous aerial vehicles [103], underwater vehicles [39] and tower cranes [18].

An overview of Dahl and Nielsen [33]'s approach, Online Scaling, is shown in Figure 2.1. The path tracking controller consists of two sub-controllers: a *path controller* and a *tracking controller*. The path controller generates a time-parameterization $s(t)$ ("online scaling") by controlling the path acceleration $\ddot{s}(t)$ online, from which a reference state $(\mathbf{q}_d(t), \dot{\mathbf{q}}_d(t)$ is computed using the relations

$$\mathbf{q}_d(t) = \mathbf{q}(s(t)), \;\; \dot{\mathbf{q}}_d(t) = \mathbf{q}'(s(t))\dot{s}(t).$$

The tracking controller receives as input the reference state $(\mathbf{q}_d(t), \dot{\mathbf{q}}_d(t))$ and computes the tracking joint torques $\boldsymbol{\tau}$ using a computed-torque tracking control law with fixed Proportional-Derivative gains [120, 79] or a PID control law [92]. Researchers have proposed many extensions to Online Scaling. Gerelli and Bianco [50, 51, 52] developed nonlinear discrete-time filters to modify time-parameterizations online that accounts for additionally joint jerk and torque rate. Recently, Bianco and Ghilardelli [15] generalized these developments to a new controller architecture that can account for generic high-order kinematic constraints.

However, the Online Scaling controllers share a common problem: There is *no guarantee* that the path controller will always be able to find feasible path accelerations online. Most papers on Online Scaling recognize this issue. Dahl and Nielsen [33] proposed to use the nominal path acceleration if there is no feasible control. Bianco and Gerelli [14] asserted that: "*since [the path control] bounds are online evaluated [...], it is not possible to guarantee [...that] a feasible solution exists [...]*". Yet, employing arbitrary controls when infeasible results in large path tracking errors, as demonstrated in the experiments in Chapter 4 and [97]. For time-optimal path tracking, this feasibility issue is even more serious, because by Pontryagin's Maximum Principle at least one joint torque limit is saturated at any time instance.

In the control literature, Model-Prediction Control (MPC) has been proposed to solve Path Tracking. In this approach, researchers formulated an augmented optimal control problem, whose state vector includes the robot's state $(\mathbf{q}, \dot{\mathbf{q}})$ and the path controller's state $(s, \dot{s})$ and control includes the robot's joint torque in addition to the path acceleration command $(\tau, \ddot{s})$ [1, 40, 41]. A known advantage of MPC controllers is the ability to account for hard constraints on state and control despite the system nonlinear dynamics. Yet, the feasibility issue that Online Scaling controllers face also exist [40]. To address this issue, Faulwasser and Findeisen [40] proposed to *design by hand* terminal sets for the optimal control instances that are solved at each time step. The resulting closed-loop dynamics can then be proven to be exponentially stable; however, designing terminal sets by hand is tedious and most likely conservative.

In Chapter 7 of this thesis, I will present an analysis of the feasibility issue that both existing approaches, Online Scaling and MPC, face. I will show that by applying robust optimization and the idea proposed in Chapter 3, one can design *automatically* terminal sets based on the expected levels of uncertainty.

2.6. Summary

In this chapter, I have given a brief description of the Time-Optimal Path Parametrization problem. Subsequently I discussed the strengths and weaknesses of current state-of-art approaches, focusing specifically on the research gap and thereby potential venue for improvement. In addition I have also discussed the applications of the Time-Optimal Path Parameterization algorithms in different use contexts, the issue of third-order constraints and finally the problem of accureate trajectory tracking of time-optimally parametrized trajectories.

Chapter 3.

Time-Optimal Path Parameterization based on Reachability Analysis

3.1. Introduction

Time-Optimal Path Parameterization (TOPP) is the problem of finding the fastest way to traverse a path in the configuration space of a robot system while respecting the system constraints [16]. This classical problem has a wide range of applications in robotics. In many industrial processes (cutting, welding, machining, 3D printing, etc.) or mobile robotic applications (driverless cars, warehouse UGVs, aircraft taxiing, etc.), the robot paths may be predefined, and optimal productivity implies tracking those paths at the highest possible speed while respecting the process and robot constraints. From a conceptual viewpoint, TOPP has been used extensively as a subroutine to kinodynamic motion planning algorithms [110, 101]. Because of its practical and theoretical importances, TOPP has received considerable attention since its inception in the 1980's.

In this chapter, we develop a new approach to TOPP based on Reachability Analysis (RA), a standard concept from control theory. The principal insight is: given an interval of squared velocities $\mathbb{I}_s$ at some position s on the path, the *reachable* set $\mathbb{I}_{s+\Delta}$ (the set of all squared velocities at the next path position that can be reached from $\mathbb{I}_s$ following admissible controls) and the *controllable* set $\mathbb{I}_{s-\Delta}$ (the set of all squared velocities at the previous path position such that there exists an admissible control leading to a velocity in $\mathbb{I}_s$) can be computed quickly and robustly by solving a few *small* Linear Programs (LPs). By recursively computing controllable sets at discretized positions on the path, one can then extract the time-optimal parameterization in time $O(mN)$, where m is the number of constraint inequalities and N the discretization grid size,

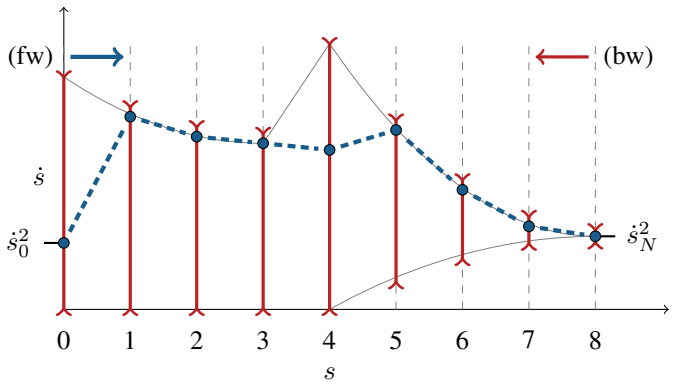


Figure 3.1.: Time-Optimal Path Parameterization by Reachability Analysis (TOPP-RA) computes the optimal parameterization in two passes. In the first pass (backward), starting from the last grid point N, the algorithm computes controllable sets (red intervals) recursively. In the second pass (forward), starting now from grid point 0, the algorithm repeatedly selects the highest controls such that resulting velocities remain inside the respective controllable sets.

see Fig. 3.1 for an illustration.

As compared to Numerical Integration (NI)-based methods, the proposed approach has a better time complexity (Actual computation time is similar for problem instances with few constraints, and becomes significantly faster for instances with > 20 constraints. Note that a robot arm with joint velocity, acceleration and torque constraints can have up to 30 constraints in the TOPP problem. Most applications have more constraints.). More importantly, the proposed method is much easier to implement and has a success rate of 100%, while state-of-the-art NI-based implementations (e.g., [99]) comprise thousands of lines of code and still report failures on hard problem instances. As compared to Convex Optimization (CO)-based methods, the proposed approach enjoys the same level of robustness and ease-of-implementation while being significantly faster.

Besides the gains in implementation robustness and performance, viewing TOPP from the proposed new perspective yields the following additional benefits:

- redundantly-actuated systems can be easily handled: there is no need to project the constraints to the plane (path acceleration $\times$ control) at each path position, as done in [54, 100];

- Admissible Velocity Propagation [101], a recent concept for kinodynamic motion planning (see Section 3.6.1 for a brief summary), can be derived "for free";

- robustness to parametric uncertainty, e.g. uncertain coefficients of friction or uncertain inertia matrices, can be obtained readily.

More details regarding the benefits as well as definitions of relevant concepts will be given in Section 3.6.

Organization of the chapter

The rest of the chapter is organized as follows. Section 3.2 formulates TOPP in a general setting. Section 3.3 applies Reachability Analysis to the path-projected dynamics. Section 3.4 presents the algorithm to compute the time-optimal path parameterization. Section 3.5 reports extensive experimental results to demonstrate the gains in robustness and performance permitted by the new approach. Section 3.6 discusses the additional benefits mentioned previously: Admissible Velocity Propagation and robustness to parametric uncertainty. Finally, Section 3.7 offers some concluding remarks and directions for future research.

3.2. Problem formulation

3.2.1. Generalized constraints

Consider a n-dof robot system, whose configuration is denoted by a n dimensional vector $\mathbf{q} \in \mathbb{R}^n$. A *geometric path* $\mathcal{P}$ in the configuration space is represented as a function $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$. We assume that $\mathbf{q}(s)$ is piece-wise $\mathcal{C}^2$-continuous. A *time parameterization* is a piece-wise $\mathcal{C}^2$, increasing scalar function $s : [0, T] \to [0, s_{\text{end}}]$, from which a *trajectory* is recovered as $\mathbf{q}(s(t))_{t\in[0,T]}$.

In this chapter, we consider *generalized second-order constraints* of the following form [54, 100]

$$\mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}(\mathbf{q}) \in \mathcal{C}(\mathbf{q}), \text{ where} \tag{3.1}$$

- $\mathbf{A}, \mathbf{B}, \mathbf{f}$ are continuous mappings from $\mathbb{R}^n$ to $\mathbb{R}^{m\times n}, \mathbb{R}^{n\times m\times n}$ and $\mathbb{R}^m$ respectively;
- $\mathcal{C}(\mathbf{q})$ is a convex polytope in $\mathbb{R}^m$.

Implementation remark 1. The above form is the most general in the TOPP literature to date and can account for many types of kinodynamic constraints, including velocity and acceleration bounds, joint torque bounds for fully- or redundantly-actuated robots [100], contact stability under Coulomb friction model [54, 24, 26].

Consider the equation of motion of a fully-actuated manipulator with bounded torques

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \tau, \tag{3.2}$$

$$\tau_i^{\min} \leq \tau_i(t) \leq \tau_i^{\max}, \ \forall i \in [1, \dots, n], \ t \in [0, T], \tag{3.3}$$

where $\mathbf{M}, \mathbf{C}, \mathbf{g}$ capture respectively the mass, Coriolis and gravitational terms. This can be put in the form of Eq. (3.1) with $\mathbf{A} := \mathbf{M}$, $\mathbf{B} := \mathbf{C}$, $\mathbf{f} := \mathbf{g}$ and

$$\mathcal{C}(\mathbf{q}) := [\tau_1^{\min}, \tau_1^{\max}] \times \cdots \times [\tau_n^{\min}, \tau_n^{\max}].$$

For closed-chain manipulators with bounded torques, one can not use Eq (3.2) and (3.3) directly to form generalized second-order constraints. One method, proposed in [100], is to use the equation of motion of an open-chain manipulator obtained by cutting the closed-chain manipulator at specific joints. In particular, let $\mathbf{q}_O$ denote the configuration of the open-chain. Suppose both chains have the same motion, [89] showed that τ is a feasible torque if

$$\begin{aligned}
&\mathbf{M}_O(\mathbf{q}_O)\ddot{\mathbf{q}}_O + \dot{\mathbf{q}}_O^\top \mathbf{C}_O(\mathbf{q}_O)\dot{\mathbf{q}}_O + \mathbf{g}_O(\mathbf{q}_O) = \boldsymbol{\tau}_O, \\
&\mathbf{S}^\top \boldsymbol{\tau} = \mathbf{W}^\top \boldsymbol{\tau}_O, \\
&\tau_i^{\min} \leq \tau_i(t) \leq \tau_i^{\max}, \ \forall i \in [1, \dots, n], \ t \in [0, T],
\end{aligned}$$

where $\mathbf{S}$ and $\mathbf{W}$ are the *sentivity matrices*, computed from the kinematics of the closed-chain manipulator and the selection of joints. See [89] for more details.

For legged robots, TOPP under contact-stability constraints with linearized friction cones was shown to be reducible to a generalized second-order constraint (3.1) [54, 100, 24].

If the friction cones are not linearized, then the set $\mathcal{C}(\mathbf{q})$ in (3.1) is still convex, but not polytopic. The developments in the present chapter that concern reachable and controllable sets (Section 3.3) are still valid in the convex, non-polytopic case. The developments on time-optimality (Section 3.4) is however only applicable to the polytopic case. ◊

Finally, we also consider first-order constraints of the form

$$\mathbf{A}^v(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}^v(\mathbf{q}) \in \mathcal{C}^v(\mathbf{q}),$$

where the coefficients are matrices of appropriate sizes and $\mathcal{C}^v(\mathbf{q})$ is a convex set. Direct velocity bounds and momentum bounds are examples of first-order constraints.

3.2.2. Projecting the constraints on the path

Differentiating successively $\mathbf{q}(s)$, one has

$$\dot{\mathbf{q}} = \mathbf{q}'\dot{s}, \quad \ddot{\mathbf{q}} = \mathbf{q}''\dot{s}^2 + \mathbf{q}'\ddot{s}, \tag{3.4}$$

where the superscript $\square'$ denotes differentiation with respect to the path parameter s, while the superscript $\dot{\square}$ denotes differentiation with respect to time. From now on, we shall refer to $s, \dot{s}, \ddot{s}$ as the position, velocity and acceleration respectively.

Substituting Eq. (3.4) to Eq. (3.1), one transforms generalized second-order constraints into constraints on $s, \dot{s}, \ddot{s}$ as follows

$$\mathbf{a}(s)\ddot{s} + \mathbf{b}(s)\dot{s}^2 + \mathbf{c}(s) \in \mathcal{C}(s), \text{ where} \tag{3.5}$$

$$\begin{aligned}
\mathbf{a}(s) &:= \mathbf{A}(\mathbf{q}(s))\mathbf{q}'(s), \\
\mathbf{b}(s) &:= \mathbf{A}(\mathbf{q}(s))\mathbf{q}''(s) + \mathbf{q}'(s)^\top \mathbf{B}(\mathbf{q}(s))\mathbf{q}'(s), \\
\mathbf{c}(s) &:= \mathbf{f}(\mathbf{q}(s)), \\
\mathcal{C}(s) &:= \mathcal{C}(\mathbf{q}(s)).
\end{aligned}$$

Similarly, first-order constraints are transformed into

$$\mathbf{a}^v(s)\dot{s} + \mathbf{b}^v(s) \in \mathcal{C}^v(s), \text{ where} \tag{3.6}$$

$$\begin{aligned}
\mathbf{a}^v(s) &:= \mathbf{A}^v(\mathbf{q}(s))\mathbf{q}'(s), \\
\mathbf{b}^v(s) &:= \mathbf{f}^v(\mathbf{q}(s)), \\
\mathcal{C}^v(s) &:= \mathcal{C}^v(\mathbf{q}(s)).
\end{aligned}$$

3.2.3. Path discretization

As in the CO-based approach, we divide the interval $[0, s_{\text{end}}]$ into N segments and $N+1$ grid points

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

Denote by u_i the constant path acceleration over the interval $[s_i, s_{i+1}]$ and by x_i the squared velocity $\dot{s}_i^2$ at s_i. One has the following relation

$$x_{i+1} = x_i + 2\Delta_i u_i, \quad i = 0 \dots N-1, \tag{3.7}$$

where $\Delta_i := s_{i+1} - s_i$. This relation is obtained by noting that

$$\frac{\mathrm{d}\dot{s}^2}{\mathrm{d}s} = 2\dot{s}\frac{\mathrm{d}\dot{s}}{\mathrm{d}s} = 2\ddot{s} = 2u.$$

In the sequel, we refer to s_i as the i-stage, u_i and x_i as respectively the control and state at the i-stage. Any sequence $x_0, u_0, \dots, x_{N-1}, u_{N-1}, x_N$ that satisfies the linear relation (3.7) is referred to as a path parameterization.

A parameterization is *admissible* if it satisfies the constraints at every points in $[0, s_{\text{end}}]$. One possible way to bring this requirement into the discrete setting is through

a *collocation* discretization scheme: for each position s_i, one evaluates the continuous constraints and requires the control and state u_i, x_i to verify

$$\mathbf{a}_i u_i + \mathbf{b}_i x_i + \mathbf{c}_i \in \mathcal{C}_i, \tag{3.8}$$

where $\mathbf{a}_i := \mathbf{a}(s_i), \mathbf{b}_i := \mathbf{b}(s_i), \mathbf{c}_i := \mathbf{c}(s_i), \mathcal{C}_i := \mathcal{C}(s_i)$.

Since constraints (3.8) are enforced only at a finite number of points, the continuous constraints (3.5) and (3.6) are not satisfied everywhere along $[0, s_{\text{end}}]$ [1]. Therefore, it is important to characterize satisfaction errors. We show in Appendix 3.8.4 that the collocation scheme has satisfaction errors of order $O(\Delta_i)$. Appendix 3.8.4 also presents another discretization scheme with satisfaction errors of order $O(\Delta_i^2)$ but which involves more variables and constraints than the collocation scheme.

3.3. Reachability Analysis of the path-projected dynamics

The key to our analysis is that the "path-projected dynamics" (3.7), (3.8) is a *discrete-time linear system* with *linear control-state inequality constraints*. This observation immediately allows us to take advantage of the set-membership control problem studied in the Model Predictive Control (MPC) literature [63, 10, 104].

3.3.1. Admissible states and controls

We first need some definitions. Denote the i-stage set of *admissible* control-state pairs by

$$\Omega_i := \{(u, x) \mid \mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathcal{C}_i\}.$$

One can see Ω_i as the projection of $\mathcal{C}_i$ on the $(\ddot{s}, \dot{s}^2)$ plane [54]. Since $\mathcal{C}_i$ is a polytope, Ω_i is a *polygon*. Algorithmically, the projection can be obtained by, e.g., the recursive expansion algorithm [22].

Next, the i-stage set of *admissible states* is the projection of Ω_i on the second axis

$$\mathcal{X}_i := \{x \mid \exists u : (u, x) \in \Omega_i\}.$$

The i-stage set of *admissible controls* given a state x is

$$\mathcal{U}_i(x) := \{u \mid (u, x) \in \Omega_i\}.$$

[1] This limitation is not specific to the proposed approach as all NI-based and CO-based algorithms require discretization at some stages.

Note that, since Ω_i is convex, both $\mathcal{X}_i$ and $\mathcal{U}_i(x)$ are *intervals*.

Classic terminologies in the TOPP literature (e.g., Maximum Velocity Curve, α and β acceleration fields) can be conveniently expressed using these definitions. See the first part of Appendix 3.8.1 for more details.

Implementation remark 2. For redundantly-actuated manipulators and legged robots with contact-stability constraints, both NI-based and CO-based algorithms must compute Ω_i at each discretized position i along the path, which is costly. Our proposed approach avoids performing this 2D projection: instead, it will only require a few 1D projections per discretization step. Furthermore, each of these 1D projections amounts to a pair of LPs and can be performed extremely quickly. ◊

3.3.2. Reachable sets

A key notion in Reachability Analysis is the i-stage reachable set.

Definition 1 (i-stage reachable set)**.** Consider a set of starting states $\mathbb{I}_0$. The *i-stage reachable set* $\mathcal{L}_i(\mathbb{I}_0)$ is the set of states $x \in \mathcal{X}_i$ such that there exist a state $x_0 \in \mathbb{I}_0$ and a sequence of admissible controls $u_0, \dots, u_{i-1}$ that steers the system from x_0 to x. ◊

To compute the i-stage reachable set, one needs the following intermediate representation.

Definition 2 (Reach set)**.** Consider a set of states $\mathbb{I}$. The *reach set* $\mathcal{R}_i(\mathbb{I})$ is the set of states $x \in \mathcal{X}_{i+1}$ such that there exist a state $\tilde{x} \in \mathbb{I}$ and an admissible control $u \in \mathcal{U}_i(\tilde{x})$ that steers the system from $\tilde{x}$ to x, i.e.,

$$x = \tilde{x} + 2\Delta_i u. \qquad \text{◊}$$

Implementation remark 3. Let us note $\Omega_i(\mathbb{I}) := \{(u, \tilde{x}) \in \Omega_i \mid \tilde{x} \in \mathbb{I}\}$. If $\mathbb{I}$ is convex, then $\Omega_i(\mathbb{I})$ is convex as it is the intersection of two convex sets. Next, $\mathcal{R}_i(\mathbb{I})$ can be seen as the intersection of the projection of $\Omega_i(\mathbb{I})$ onto a line and the interval $\mathcal{X}_{i+1}$. Thus, $\mathcal{R}_i(\mathbb{I})$ is an interval, hence defined by its lower and upper bounds (x^-, x^+), which can be computed as follows

$$x^- := \min_{(u,\tilde{x}) \in \Omega_i(\mathbb{I}),\ x^- \in \mathcal{X}_{i+1}} (\tilde{x} + 2\Delta_i u),$$

$$x^+ := \max_{(u,\tilde{x}) \in \Omega_i(\mathbb{I}),\ x^+ \in \mathcal{X}_{i+1}} (\tilde{x} + 2\Delta_i u).$$

Since $\Omega_i(\mathbb{I})$ is a polygon, the above equations constitute two LPs. Note finally that

there is no need to compute explicitly $\Omega_i(\mathbb{I})$, since one can write directly

$$x^+ := \max_{(u,\tilde{x})\in\mathbb{R}^2} (\tilde{x} + 2\Delta_i u),$$
$$\text{subject to: } \mathbf{a}_i u + \mathbf{b}_i \tilde{x} + \mathbf{c}_i \in \mathcal{C}_i, \tilde{x} \in \mathbb{I} \text{ and } x^+ \in \mathcal{X}_{i+1},$$

and similarly for x^-. ◊

The i-stage reachable set can be recursively computed by

$$\begin{aligned} \mathcal{L}_0(\mathbb{I}_0) &= \mathbb{I}_0 \cap \mathcal{X}_0, \\ \mathcal{L}_i(\mathbb{I}_0) &= \mathcal{R}_{i-1}(\mathcal{L}_{i-1}(\mathbb{I}_0)). \end{aligned} \tag{3.9}$$

Implementation remark 4. If $\mathbb{I}_0$ is an interval, then by recursion and by application of Implementation remark 3, all the $\mathcal{L}_i$ are intervals. Each step of the recursion requires solving two LPs for computing $\mathcal{R}_{i-1}(\mathcal{L}_{i-1}(\mathbb{I}_0))$. Therefore, $\mathcal{L}_i$ can be computed by solving $2i + 2$ LPs. ◊

The i-stage reachable set may be empty, which implies that the system cannot evolve without violating constraints: the path is not time-parameterizable. One can also note that

$$\mathcal{L}_i(\mathbb{I}_0) = \emptyset \implies \forall j \geq i,\ \mathcal{L}_j(\mathbb{I}_0) = \emptyset.$$

3.3.3. Controllable sets

Controllability is the dual notion of reachability, as made clear by the following definitions.

Definition 3 (i-stage controllable set)**.** Consider a set of desired ending states $\mathbb{I}_N$. The *i-stage controllable set* $\mathcal{K}_i(\mathbb{I}_N)$ is the set of states $x \in \mathcal{X}_i$ such that there exist a state $x_N \in \mathbb{I}_N$ and a sequence of admissible controls $u_i, \ldots, u_{N-1}$ that steers the system from x to x_N. ◊

The dual notion of "reach set" is that of "one-step" set.

Definition 4 (One-step set)**.** Consider a set of states $\mathbb{I}$. The *one-step set* $\mathcal{Q}_i(\mathbb{I})$ is the set of states $x \in \mathcal{X}_i$ such that there exist a state $\tilde{x} \in \mathbb{I}$ and an admissible control $u \in \mathcal{U}_i(x)$ that steers the system from x to $\tilde{x}$, i.e.

$$\tilde{x} = x + 2\Delta_i u.$$ ◊

The i-stage controllable set can now be computed recursively by

$$\begin{aligned} \mathcal{K}_N(\mathbb{I}_N) &= \mathbb{I}_N \cap \mathcal{X}_N, \\ \mathcal{K}_i(\mathbb{I}_N) &= \mathcal{Q}_i(\mathcal{K}_{i+1}(\mathbb{I}_N)). \end{aligned} \tag{3.10}$$

Implementation remark 5. Similar to Implementation remark 4, every one-step set $\mathcal{Q}_i(\mathbb{I})$ is an interval, whose lower and upper bounds (x^-, x^+) can be computed as follows

$$x^+ := \max_{(u,x)\in\mathbb{R}^2} x,$$
$$\text{subject to: } \mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathcal{C}_i \text{ and } x + 2\Delta_i u \in \mathbb{I},$$

and similarly for x^-. Thus, computing the i-stage controllable set will require solving $2(N-i)+2$ LPs. ◊

The i-stage controllable set may be empty, in that case, the path is not time-parameterizable. One also has

$$\mathcal{K}_i(\mathbb{I}_N) = \emptyset \implies \forall j \leq i,\ \mathcal{K}_j(\mathbb{I}_N) = \emptyset.$$

3.4. TOPP by Reachability Analysis

3.4.1. Algorithm

Armed with the notions of reachable and controllable sets, we can now proceed to solving the TOPP problem. The Reachability-Analysis-based TOPP algorithm (TOPP-RA) is given in Algorithm 1 below and illustrated in Fig. 3.1.

Algorithm 1: TOPP-RA

Input : Path $\mathcal{P}$, starting and ending velocities $\dot{s}_0, \dot{s}_N$
Output: Parameterization $x_0^*, u_0^*, \ldots, u_{N-1}^*, x_N^*$

`/* Backward pass: compute the controllable sets */`

$\mathcal{K}_N := \{\dot{s}_N^2\}$
for $i \in [N-1 \ldots 0]$ **do**
  $\mathcal{K}_i := \mathcal{Q}_i(\mathcal{K}_{i+1})$
if $\mathcal{K}_0 = \emptyset$ *or* $\dot{s}_0^2 \notin \mathcal{K}_0$ **then**
  return `Infeasible`

`/* Forward pass: select controls greedily */`

$x_0^* := \dot{s}_0^2$
for $i \in [0 \ldots N-1]$ **do**
  $u_i^* :=$ max u, subject to: $x_i^* + 2\Delta_i u \in \mathcal{K}_{i+1}$ and $(u, x_i^*) \in \Omega_i$
  $x_{i+1}^* := x_i^* + 2\Delta_i u_i^*$

The algorithm proceeds in two passes. The first pass goes backward: it recursively computes the controllable sets $\mathcal{K}_i(\{\dot{s}_N^2\})$ given the desired ending velocity $\dot{s}_N$, as described in Section 3.3.3. If any of the controllable sets is empty or if the starting state $\dot{s}_0^2$ is not contained in the 0-stage controllable set, then the algorithm reports failure.

Otherwise, the algorithm proceeds to a second, forward, pass. Here, the optimal states and controls are constructed *greedily*: at each stage i, the highest admissible

control such that the resulting next state belongs to the $(i+1)$-stage controllable set is selected.

Note that one can construct a "dual version" of TOPP-RA as follows: (i) in a forward pass, recursively compute the i-stage reachable sets, $i \in [0, \dots, N]$; (ii) in a backward pass, greedily select, at stage i, the lowest control such that the previous state belongs to the $(i-1)$-stage reachable set.

In the following sections, we show the correctness and optimality of the algorithm and give a detailed complexity analysis.

3.4.2. Correctness of TOPP-RA

We show that TOPP-RA is correct in the sense of the following theorem.

Theorem 1. *Consider a discretized TOPP instance. TOPP-RA returns an admissible parameterization solving that instance whenever one exists, and reports* `Infeasible` *otherwise.*

Proof. (1) We first show that, if TOPP-RA reports `Infeasible`, then the instance is indeed not parameterizable. By contradiction, assume that there exists an admissible parameterization $\dot{s}_0^2 = x_0, u_0, \dots, u_{N-1}, x_N = \dot{s}_N^2$. We now show by backward induction on i that $\mathcal{K}_i$ contains at least x_i.

Initialization: $\mathcal{K}_N$ contains x_N by construction.

Induction: Assume that $\mathcal{K}_i$ contains x_i. Since the parameterization is admissible, one has $x_i = x_{i-1} + 2\Delta_i u_{i-1}$ and $(u_{i-1}, x_{i-1}) \in \Omega_{i-1}$. By definition of the controllable sets, $x_{i-1} \in \mathcal{K}_{i-1}$.

We have thus shown that none of the $\mathcal{K}_i$ is empty and that $\mathcal{K}_0$ contains at least $x_0 = \dot{s}_0^2$, which implies that TOPP-RA cannot report `Infeasible`.

(2) Assume now that TOPP-RA returns a sequence $(x_0^*, u_0^*, \dots, u_{N-1}^*, x_N^*)$. One can easily show by forward induction on i that the sequence indeed constitutes an admissible parameterization that solves the instance. □

3.4.3. Asymptotic optimality of TOPP-RA

We show the following result: as the discretization step size goes to zero, the traversal time of the parameterization returned by TOPP-RA converges to the optimal value.

Unsurprisingly, the main difficulty in proving asymptotic optimality comes from the existence of zero-inertia points [94, 112, 99]. Note however that this difficulty does not affect the robustness or the correctness of the algorithm.

To avoid too many technicalities, we make the following assumption.

Assumption 1 (and definition)**.** There exist piece-wise $\mathcal{C}^1$-continuous functions $\tilde{\mathbf{a}}(s)_{s\in[0,1]}$, $\tilde{\mathbf{b}}(s)_{s\in[0,1]}, \tilde{\mathbf{c}}(s)_{s\in[0,1]}$ such that for all $i \in \{0, \dots, N\}$, the set of admissible control-state pairs is given by

$$\Omega_i = \{(u, x) \mid u\tilde{\mathbf{a}}(s_i) + x\tilde{\mathbf{b}}(s_i) + \tilde{\mathbf{c}}(s_i) \leq 0\}.$$

Augment $\tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}}$ into $\bar{\mathbf{a}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}$ by adding two inequalities that express the condition $x + 2\Delta_i u \in \mathcal{K}_{i+1}$. The set of admissible *and controllable* control-state pairs is then given by

$$\Omega_i \cap (\mathbb{R} \times \mathcal{K}_i) = \{(u, x) \mid u\bar{\mathbf{a}}(s_i) + x\bar{\mathbf{b}}(s_i) + \bar{\mathbf{c}}(s_i) \leq 0\}. \qquad \Diamond$$

The above assumption is easily verified in the canonical case of a fully-actuated manipulator subject to torque bounds tracking a smooth path. It allows us to next conveniently define zero-inertia points.

Definition 5 (Zero-inertia points)**.** A point $s^\bullet$ constitutes a zero-inertia point if there is a constraint k such that $\bar{\mathbf{a}}(s^\bullet)[k] = 0$. $\Diamond$

We have the following theorem, whose proof is given in Appendix 3.8.2 (to simplify the notations, we consider uniform step sizes $\Delta_0 = \cdots = \Delta_{N-1} = \Delta$).

Theorem 2. *Consider a TOPP instance* without zero-inertia points. *There exists a* Δ_{thr} *such that if* $\Delta < \Delta_{\text{thr}}$*, then the parameterization returned by TOPP-RA is optimal.*

The key hypothesis of this theorem is that there is no zero-inertia points. In practice, however, zero-inertia points are unavoidable and in fact constitute the most common type of switch points [99]. The next theorem, whose proof is given in Appendix 3.8.3, establishes that the sub-optimality gap converges to zero with step size.

Theorem 3. *Consider a TOPP instance with a zero-inertia point at* $s^\bullet$*. Denote by* J^* *the cost of the parameterization returned by TOPP-RA* $\sum_{i=0}^{N+1} \frac{\Delta}{\sqrt{x_i^*}}$ *and by* $J^\dagger$ *the minimum cost at the same step size. Then one has*

$$J^* - J^\dagger = O(\Delta).$$

This theorem implies that, by reducing the step size, the cost of the parameterization returned by TOPP-RA can be made arbitrarily close to the minimum cost.

3.4.4. Complexity analysis

We now perform a complexity analysis of TOPP-RA and compare it with NI and CO-based methods. For simplicity, we shall restrict the discussion to fully-actuated manipulators (the redundantly-actuated case actually brings an additional advantage to TOPP-RA, see Implementation remark 3).

Assume that there are m constraint inequalities and that the path discretization grid size is N. As a large part of the computation time is devoted to solving LPs, we need a good estimate of the practical complexity of this operation. Consider a LP with ν optimization variables and m inequality constraints. Different LP methods (ellipsoidal, simplex, active sets, etc.) have different complexities. For the purpose of this section, we consider the best *practical* complexity, which is realized by the simplex method, in $O(\nu^2 m)$ [20].

- *TOPP-RA:* The LPs considered here have 2 variables and $m + 2$ inequalities. Since one needs to solve $3N$ such LPs, the complexity of TOPP-RA is $O(mN)$.
- *Numerical integration approach:* The dominant component of this approach, in terms of time complexity, is the computation of the Maximum Velocity Curve (MVC). In most TOPP-NI implementations to date, the MVC is computed, at each discretized path position, by solving $O(m^2)$ second-order polynomials [16, 94, 119, 112, 99], which results in an overall complexity of $O(m^2 N)$.
- *Convex optimization approach:* This approach formulates the TOPP problem as a single large convex optimization program with $O(N)$ variables and $O(mN)$ inequality constraints. In the fastest implementation we know of, the author solves the convex optimization problem by solving a sequence of linear programs (SLP) with the same number of variables and inequalities [54]. Thus, the time complexity of this approach is $O(KmN^3)$, where K is the number of SLP iterations.

This analysis shows that TOPP-RA has the best theoretical complexity. The next section experimentally assesses this observation.

3.5. Simulations

We implement TOPP-RA in Python. All LPs are solved with `qpOASES` [42]. Experiments were performed on a machine running Ubuntu with an Intel i7-4770(8) 3.9GHz CPU and 8Gb RAM. The implementation and test cases are available at https://github.com/hungpham2511/toppra.

3.5.1. Experiment 1: Pure joint velocity and acceleration bounds

In this experiment, we compare TOPP-RA against TOPP-NI – the fastest known implementation of TOPP, which is based on the NI approach [99]. For simplicity, we

consider pure joint velocity and acceleration bounds, which involve the same difficulty as any other types of kinodynamic constraints, as far as TOPP is concerned.

Effect of the number of constraint inequalities

We considered random geometric paths with varying dimensions $n \in [2, 60]$. Each path was generated as follows: 5 random waypoints were sampled, then interpolated by cubic spline interpolation. For each path, velocity and acceleration bounds were also randomly chosen such that the bounds contain zero. This ensures that all generated TOPP instances are feasible. Each instance thus has $m = 2n + 2$ constraint inequalities: $2n$ inequalities corresponding to acceleration bounds (no pruning was applied, contrary to [54]) and 2 inequalities corresponding to velocity bounds (the joint velocity bounds could be immediately pruned into one lower and one upper bound on $\dot{s}^2$). According to the complexity analysis of Section 3.4.4, we consider the number of inequalities, rather than the dimension, as independent variable. Finally, the discretization grid size was chosen as $N = 500$.

Fig. 3.2 shows the time-parameterizations and the resulting trajectories produced by TOPP-RA and TOPP-NI on an instance with $(n = 6, m = 14)$. One can observe that the two algorithms produced nearly identical results, hinting at the correctness of TOPP-RA.

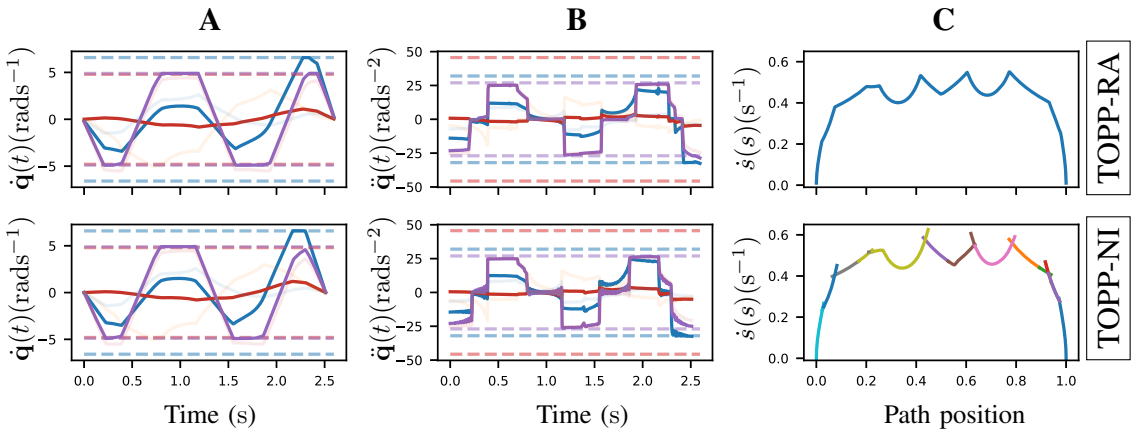


Figure 3.2.: Time-optimal parameterization of a 6-dof path under velocity and acceleration bounds ($m = 14$ constraint inequalities and $N = 500$ grid points). TOPP-RA and TOPP-NI produce nearly identical results. **(A)**: joint velocities. **(B)**: joint accelerations. **(C)**: velocity profiles in the $(s, \dot{s})$ plane. Note the small chattering in the joint accelerations produced by TOPP-NI, which is an artifact of the integration process. This chattering is absent from the TOPP-RA profiles.

Fig. 3.3 shows the computation time for TOPP-RA and TOPP-NI, excluding the "setup" and "extract trajectory" steps (which takes much longer in TOPP-NI than in

TOPP-RA). The experimental results confirm our theoretical analysis in that the complexity of TOPP-RA is in linear in m while that of TOPP-NI is quadratic in m. In terms of actual computation time, TOPP-RA becomes faster than TOPP-NI as soon as $m \geq 22$.

Perhaps even more importantly than mere computation time, TOPP-RA was extremely robust: it maintained 100% success rate over all instances, while TOPP-NI struggled with instances with many inequality constraints ($m \geq 40$), see Fig. 3.4. Since all TOPP instances considered here were feasible, an algorithm failed when it failed to return a parameterization.

Table 3.1 reports the different components of the computation time. In addition to TOPP-RA and TOPP-NI, we considered TOPP-RA-intp. This variant of TOPP-RA employs the first-order interpolation scheme (see Appendix 3.8.4) to discretize the constraints, instead of the collocation scheme introduced in Section 3.2.3.

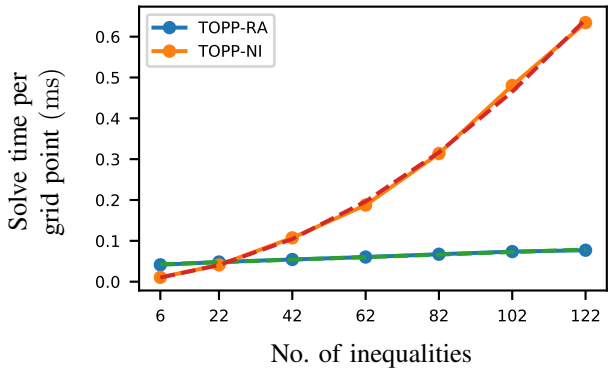


Figure 3.3.: Solve time per grid point of TOPP-RA (solid blue) and TOPP-NI (solid orange), excluding the "setup" and "extract trajectory" steps, as a function of the number of constraint inequalities. Confirming our theoretical complexity analysis, time complexity of TOPP-RA is linear in the number of constraint inequalities m (linear fit in dashed green), while that of TOPP-NI is quadratic in m (quadratic fit in dashed red). In terms of actual computation time, TOPP-RA becomes faster than TOPP-NI as soon as $m \geq 30$.

Effect of discretization grid size

Discretization grid size (or its inverse, discretization step) is an important parameter for both TOPP-RA and TOPP-NI as it affects running time, success rate and solution quality, as measured by constraint satisfaction error and sub-optimality. Here, we assess the effect of grid size on *success rate* and *solution quality*. Remark that, based on our complexity analysis in Section 3.4.4, running time depends linearly on grid size in both algorithms.

Table 3.1.: Breakdown of TOPP-RA, TOPP-RA-intp and TOPP-NI total computation time to parameterize a path discretized with $N = 500$ grid points, subject to $m = 30$ inequalities.

	Time (ms)		
	TOPP-RA	TOPP-RA-intp	TOPP-NI
setup	1.0	1.5	123.6
solve TOPP	26.1	29.1	28.3
backward pass	16.5	19.9	
forward pass	9.6	9.2	
extract trajectory	2.7	2.7	303.4
total	29.8	33.3	455.3

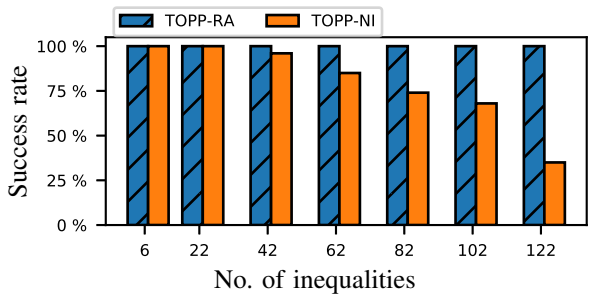


Figure 3.4.: Success rate for TOPP-RA and TOPP-NI. TOPP-RA enjoys consistently 100% success rate while TOPP-NI reports failure for more complex problem instances ($m \geq 40$).

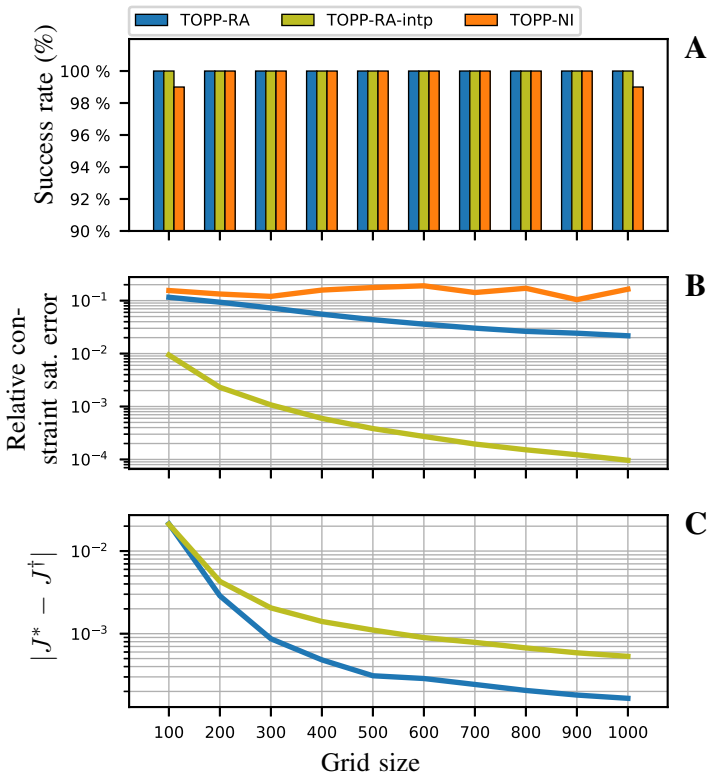


Figure 3.5.: **(A)**: effect of grid size on success rate. **(B)**: effect of grid size on relative constraint satisfaction error, defined as the ratio between the error and the respective bound. TOPP-RA-intp returned solutions that are orders of magnitude better than TOPP-RA and TOPP-NI. **(C)**: effect of grid size on difference between the average solution's cost and the optimal cost, which is approximated by solving TOPP-RA-intp with $N = 10000$. Solutions produced by TOPP-RA-intp had higher costs than those produced by TOPP-RA as those instances were more highly constrainted.

We considered different grid sizes $N \in [100, 1000]$. For each grid size, we generated and solved 100 random TOPP instances; each instance consists of a random path with $n = 14$ subject to random kinematic constraints, as in the previous experiment. Fig. 3.5-**A** shows success rates versus grid sizes. One can observe that TOPP-RA and TOPP-RA-intp maintained 100% success rate across all grid sizes, while TOPP-NI reported two failures at $N = 100$ and $N = 1000$.

Next, to measure the effect of grid size on solution quality, we looked at the *relative greatest constraint satisfaction errors*, defined as the ratio between the errors, whose definition is given in Appendix 3.8.4, and the respective bounds. For each instance, we sampled the resulting trajectories at $1\,\mathrm{ms}$ and computed the greatest constraint satisfaction errors by comparing the sampled joint accelerations and velocities to their respective bounds. Then, we averaged instances with the same grid size to obtain the average error for each N.

Fig. 3.5-**B** shows the average relative greatest constraint satisfaction errors of the

three algorithms with respect to grid size. One can observe that TOPP-RA and TOPP-NI had constraint satisfaction errors of the same order of magnitude for $N < 500$, while TOPP-RA demonstrated better quality for $N \geq 500$. TOPP-RA-intp produced solutions with much higher quality. This result confirms our error analysis of different discretization schemes in Appendix 3.8.4 and demonstrates that the interpolation discretization scheme is better than the collocation scheme whenever solution quality is concerned.

Fig. 3.5-**C** shows the average difference between the costs of solutions returned by TOPP-RA and TOPP-RA-intp with the *true* optimal cost, which was approximated by running TOPP-RA-intp with grid size $N = 10000$. One can observe that both algorithms are asymptotically optimal. Even more importantly, the differences were relatively small even at coarse grid sizes.

3.5.2. Experiment 2: Legged robot in multi-contact

Here we consider the TOPP problem for a 50-dof legged robot subject to joint torque bounds and contact stability constraints with linearized friction cones.

Formulation

We now give a brief description of our formulation, for more details, refer to [54, 100]. Let $\mathbf{w}_i$ denote the net contact wrench (force-torque pair) exerted on the robot by the i-th contact at point $\mathbf{p}_i$. Using the linearized friction cone, one obtains the set of feasible wrenches as a polyhedral cone

$$\{\mathbf{w}_i \mid \mathbf{F}_i \mathbf{w}_i \leq 0\},$$

for some matrix $\mathbf{F}_i$. This matrix can be found using the Cone Double Description method [44, 26]. Combining with the equation governing rigid-body dynamics, one obtains the full dynamic feasibility constraint as follows

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) &= \boldsymbol{\tau} + \sum_{i=1,2} \mathbf{J}_i(\mathbf{q})^\top \mathbf{w}_i, \\ \mathbf{F}_i \mathbf{w}_i &\leq 0, \\ \boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} &\leq \tau_{\max}, \end{aligned}$$

where $\mathbf{J}_i(\mathbf{q})$ is the wrench Jacobian. The convex set $\mathcal{C}(\mathbf{q})$ in Eq. (3.1) can now be identified as a multi-dimensional polyhedron.

We considered a simple swaying motion: the robot stands with both feet lie flat on two uneven steps and shift its body back and forth, see Fig. 3.6. The coefficient of fric-

tion was set to $\mu = 0.5$. Start and end path velocities were set to zero. Discretization grid size was $N = 100$. The number of constraint inequalities was $m = 242$.

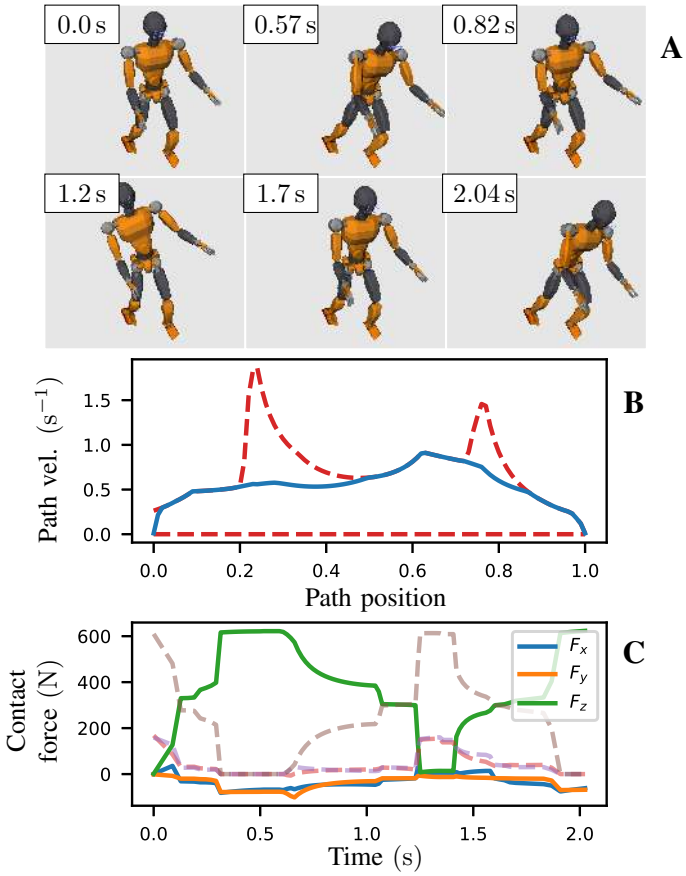


Figure 3.6.: Time-parameterization of a legged robot trajectory under joint torque bounds and multi-contact friction constraints. **(A)**: Snapshots of the re-timed motion. **(B)**: Optimal velocity profile computed by TOPP-RA (blue) and upper and lower limits of the controllable sets (dashed red). **(C)**: The optimal joint torques and contact forces are obtained "for free" as slack variables of the optimization programs solved in the forward pass. Net contact forces for the left foot are shown in colors and those for the right foot are shown in transparent lines.

Results

Excluding computations of dynamic quantities, TOPP-RA took 267 ms to solve the TOPP instance. The parameterization is shown in Fig. 3.6. Computation time is given in Table 3.2.

Compared to TOPP-NI and TOPP-CO, TOPP-RA had significantly better computation time, chiefly because both existing methods require an expensive polytopic projection step. Indeed, [54] reported projection time of 2.4 s for a similar sized problem, which is significantly more expensive than TOPP-RA computation time. Notice that

in [54], computing the parameterization takes an addition $2.46\,\mathrm{s}$ which leads to a total computation time of $4.86\,\mathrm{s}$.

To make a more accurate comparison, we implement the pipeline given in [100] to solve the TOPP instance

1. project the constraint polyhedron $\mathcal{C}_i$ onto the path using Bretl's polygon recursive expansion algorithm [22];

2. parameterize the resulting problem using TOPP-NI.

This pipeline turned out to be much slower than TOPP-RA. We found that the number of LPs the projection step solved was nearly 8 times the number of LPs solved by TOPP-RA (which is fixed at $3N = 300$). For a more detailed comparison of computation time and parameters of the LPs, refer to Table 3.2.

Obtaining joint torques and contact forces "for free"

Another interesting feature of TOPP-RA is that the algorithm can optimize and obtain joint torques and contact forces "for free" without additional processing. Concretely, since joint torques and contact forces are slack variables, one can simply store the optimal slack variable at each step and obtain a trajectory of feasible torques and forces. To optimize the torques and forces, we can modify the i-th step of the forward pass to solve the following quadratic program (QP)

$$\begin{aligned} \min \quad & -u + \epsilon \|(\mathbf{w}, \boldsymbol{\tau})\|_2^2 \\ \text{s.t.} \quad & x = x_i \\ & (u, x) \in \Omega_i \\ & x + 2\Delta_i u \in K_{i+1}, \end{aligned}$$

where ϵ is a positive scalar. Recall that $\mathbf{w}$ denotes the contact wrenches and $\boldsymbol{\tau}$ denotes the joint torques. Figure 3.6's lower plot shows computed contact wrench for the left leg. We note that both existing approaches, TOPP-NI and TOPP-CO are not able to produce joint torques and contact forces readily as they "flatten" the constraint polygon in the projection step.

In fact, the above formulation suggests that time-optimality is simply a specific objective cost function (linear) of the more general family of quadratic objectives. Therefore, one can in principle depart from time-optimality in favor of more realistic objective such as minimizing torque while maintaining a certain nominal velocity

x_{norm} as follow

$$\begin{aligned} \min \quad & \|x_i + 2\Delta_i u - x_{\text{norm}}\|_2^2 + \epsilon\|(\mathbf{w}, \boldsymbol{\tau})\|_2^2 \\ \text{s.t.} \quad & x = x_i \\ & (u, x) \in \Omega_i \\ & x + 2\Delta_i u \in K_{i+1}. \end{aligned}$$

Finally, we observed that the choice of path discretization scheme has noticeable effects on both computational cost and quality of the result. In general, TOPP-RA-intp produced smoother trajectories and better (lower) constraint satisfaction error at the cost of longer computation time. On the other hand, TOPP-RA was faster but produced trajectories with jitters [2] near dynamic singularities [99] and had higher constraint satisfaction error.

Table 3.2.: Computation time (ms) and internal parameters comparison between TOPP-RA, TOPP-RA-intp and TOPP-NI in Experiment 2.

	TOPP-RA	TOPP-RA-intp	TOPP-NI
	Time (ms)		
comp. dynamic quantities	181.6	193.6	281.6
polytopic projection	0.0	0.0	3671.8
solve TOPP	267.0	1619.0	335.0
extract trajectory	3.0	3.0	210.0
total	451.6	1815.6	4497.8
	Parameters		
joint torques / contact forces avail.	yes	yes	no
No. of LP(s) solved	300	300	2110
No. of variables	64	126	64
No. of constraints	242	476	242
Constraints sat. error	$O(\Delta)$	$O(\Delta^2)$	$O(\Delta)$

3.6. Additional benefits of TOPP by Reachability Analysis

We now elaborate on the additional benefits provided by the Reachability Analysis approach to TOPP.

[2] Experiments prove that singularities do not cause failures for TOPP-RA. The jitters can usually be removed easily,.i.e, using cubic splines interpolation to smooth the parametrization locally around jitters.

3.6.1. Admissible Velocity Propagation

Admissible Velocity Propagation (AVP) is a recent concept for kinodynamic motion planning [101]. Specifically, given a path and an initial interval of velocities, AVP returns exactly the interval of all the velocities the system can reach after traversing the path while respecting the system kinodynamic constraints. Combined with existing *geometric* path planners, such as RRT [65], this can be advantageously used for *kinodynamic* motion planning: at each tree extension in the configuration space, AVP can be used to guarantee the eventual existence of admissible path parameterizations.

Suppose that the initial velocity interval is $\mathbb{I}_0$. It can be immediately seen that, what is computed by AVP is exactly the reachable set $\mathcal{L}_N(\mathbb{I}_0)$ (cf. Section 3.3.2). Furthermore, what is computed by AVP-Backward [70] given a desired final velocity interval $\mathbb{I}_N$ is exactly the controllable set $\mathcal{K}_0(\mathbb{I}_N)$ (cf. Section 3.3.3). In terms of complexity, $\mathcal{R}_N(\mathbb{I}_0)$ and $\mathcal{K}_0(\mathbb{I}_N)$ can be found by solving respectively $2N + 2$ and $2N + 2$ LPs. We have thus re-derived the concept of AVP at no cost.

3.6.2. Robustness to parametric uncertainty

In most works dedicated to TOPP, including the development of the present chapter up to this point, the parameters appearing in the dynamics equations and in the constraints are supposed to be exactly known. In reality, those parameters, which include inertia matrices or payloads in robot manipulators, or feet positions or friction coefficients in legged robots, are only known up to some precision. An admissible parameterization for the nominal values of the parameters might not be admissible for the actual values, and the probability of constraints violation is even higher in the *optimal* parameterization, which saturates at least one constraint at any moment in time.

TOPP-RA provides a natural way to handle parametric uncertainties. Assume that constraints appear in the following form

$$\begin{aligned} &\mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathcal{C}_i, \\ &\forall (\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i, \mathcal{C}_i) \in \mathcal{E}_i, \end{aligned} \tag{3.11}$$

where $\mathcal{E}_i$ contains all the possible values that the parameters might take at path position s_i.

Implementation remark 6. Consider for instance the manipulator with torque bounds of equation (3.2). Suppose that, at path position i, the inertia matrix is uncertain, i.e., that it might take any values $\mathbf{M}_i \in B(\mathbf{M}_i^{\text{nominal}}, \epsilon)$, where $B(\mathbf{M}_i^{\text{nominal}}, \epsilon)$ denotes the ball of radius ϵ centered around $\mathbf{M}_i^{\text{nominal}}$ for the max norm. Then, the first component of $\mathcal{E}_i$ is given by $\{\mathbf{M}_i \mathbf{q}'(s_i) \mid \mathbf{M}_i \in B(\mathbf{M}_i^{\text{nominal}}, \epsilon)\}$, which is a convex set.

In legged robots, uncertainties on feet positions or on friction coefficients can be encoded into a "set of sets", in which $\mathcal{C}_i$ can take values. ◊

TOPP-RA can handle this situation by suitably modifying its two passes. Before presenting the modifications, we first give some definitions. Denote the i-stage set of *robust admissible* control-state pairs by

$$\widehat{\Omega}_i := \{(u, x) \mid \text{Eq. (3.11) holds}\}.$$

The sets of robust admissible states $\widehat{\mathcal{X}}_i$ and robust admissible controls $\widehat{\mathcal{U}}_i(x)$ can be defined as in Section 3.3.1.

In the backward pass, TOPP-RA computes the *robust controllable sets*, whose definition is given below.

Definition 6 (i-stage *robust* controllable set)**.** Consider a set of desired ending states $\mathbb{I}_N$. The *i-stage robust controllable set* $\widehat{\mathcal{K}}_i(\mathbb{I}_N)$ is the set states $x \in \widehat{\mathcal{X}}_i$ such that there exists a state $x_N \in \mathbb{I}_N$ and a sequence of *robust admissible controls* $u_i, \dots, u_{N-1}$ that steers the system from x to x_N. ◊

To compute the robust controllable sets, one needs the robust one-step set.

Definition 7 (*Robust* one-step set)**.** Consider a set of states $\mathbb{I}$. The *robust one-step set* $\widehat{\mathcal{Q}}_i(\mathbb{I})$ is the set of states $x \in \widehat{\mathcal{X}}_i$ such that there exists a state $\tilde{x} \in \mathbb{I}$ and a robust admissible control $u \in \widehat{\mathcal{U}}_i(x)$ that steers the system from x to $\tilde{x}$. ◊

Finally, in the forward pass, the algorithm selected the *greatest* robust admissible control at each stage.

Implementation remark 7. Computing the robust one-step set and the greatest robust admissible control involves solving LPs with uncertain constraints of the form (3.11). In the mathematical optimization literature, they are known as "Robust Linear Programs", and specific methods have been developed to handle them efficiently, when the robust constraints are [6]

1. polyhedra;
2. ellipsoids;
3. Conic Quadratic re-presentable (CQr) sets.

The first case can be treated as normal LPs with appropriate slack variables, while the last two cases are explicit Conic Quadratic Program (CQP). For more information on this conversion, refer to the first and second chapters of [6]. ◊

3.7. Summary

We have presented a new approach to solve the Time-Optimal Path Parameterization (TOPP) problem based on Reachability Analysis (TOPP-RA). The key insight is to compute, in a first pass, the sets of controllable states, for which admissible controls allowing to reach the goal are guaranteed to exist. Time-optimality can then be obtained, in a second pass, by a greedy strategy. We have shown, through theoretical analyses and extensive experiments, that the proposed algorithm is extremely robust (100% success rate) and competitive in terms of computation time as compared to the fastest known TOPP implementation [99], and produces solutions with high quality. Finally, the new approach yields additional benefits: no need for polytopic projection in the redundantly-actuated case, Admissible Velocity Projection, and robustness to parameter uncertainty.

A recognized disadvantage of the classical TOPP formulation is that the time-optimal trajectory contains hard acceleration switches, corresponding to infinite jerks. Solving TOPP subject to jerk bounds, however, is not possible using the CO-based approach as the problem becomes non-convex [125]. Some prior works proposed to either extend the NI-based approach [124, 96] or to represent the parameterization as a spline and optimize directly over the parameter space [28, 91]. Exploring how Reachability Analysis can be extended to handle jerk bounds is another direction of our future research.

Similar to the CO-based approach, Reachability Analysis can only be applied to instances with convex constraints [125]. Yet in practice, it is often desirable to consider in addition non-convex constraints, such as joint torque bounds with viscous friction effect. Extending Reachability Analysis to handle non-convex constraints is another important research question.

3.8. Additional remarks and proofs

3.8.1. Relation between TOPP-RA and TOPP-NI

TOPP-RA and TOPP-NI are subtly related: they compute the same velocity profiles, but in different orders. Let us first recall some terminologies from the literature of the NI-based approach to TOPP (see [99] for more details):

- Maximum Velocity Curve (MVC): mapping from path position to the highest dynamically feasible path velocity;

- integrate forward (or backward) following α (or β): for each tuple $(s, \dot{s})$, $\alpha(s, \dot{s})$

and $\beta(s, \dot{s})$ are the smallest and largest controls respectively; forward and backward integrations are done following the respective controls;

- $\alpha \to \beta$ switch points: there are three kinds of $\alpha \to \beta$ switch points: tangent, singular, discontinuous;

- $\dot{s}_{\text{beg}}, \dot{s}_{\text{end}}$: starting and ending velocities at s_{beg} and s_{end}.

Note that α, β functions recalled above are different from the functions defined in Definition 8 (Appendix 3.8.2). The formers maximize over the set of feasible states while the laters maximize over the set of feasible *and controllable* states.

TOPP-NI proceeds as follows:

1. determine the $\alpha \to \beta$ switch points;

2. from each $\alpha \to \beta$ switch point, integrate forward following β and backward following α to obtain the Limiting Curves (LCs);

3. take the lowest value of the LCs at each position to form the Concatenated Limiting Curve (CLC);

4. from $(0, \dot{s}_{\text{beg}})$ integrate forward following β; from $(s_{\text{end}}, \dot{s}_{\text{end}})$ integrate backward following α until their intersections with the CLC; then return the combined $\beta - \text{CLC} - \alpha$ profile.

We now rearrange the above steps into a backward pass and a forward pass in order to highlight the relation with TOPP-RA.

Backward pass

1) determine the $\alpha \to \beta$ switch points;

2a) from each $\alpha \to \beta$ switch point, integrate backward following α to obtain the Backward Limiting Curves (BLCs);

2b) from the point $(s_{\text{end}}, \dot{s}_{\text{end}})$ integrate backward following α to obtain the last BLC;

3) take the lowest value of the BLC's *and the* MVC to form the upper boundary of the controllable sets.

Forward pass

4a) set the current point to the point $(s_{\text{beg}}, \dot{s}_{\text{beg}})$;

4*b*) repeat until the current point is the point $(s_{\text{end}}, \dot{s}_{\text{end}})$, from the current point integrate forward following β until hitting a BLC, set the corresponding switch point as the new current point.

See Fig. 3.7 for a visualization of the rearrangement.

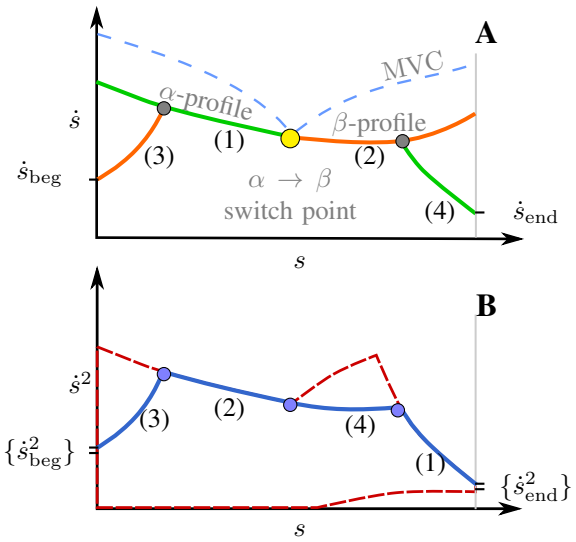


Figure 3.7.: TOPP-NI **(A)** and TOPP-RA **(B)** compute the time-optimal path parameterization by creating similar profiles in different ordering. (1,2,3,4) is the order in which TOPP-RA computes the profiles.

The key idea in this rearrangement is to not compute β profiles immediately for each switch point, but delay until needed. The resulting algorithm is almost identical to TOPP-RA except for the following points

- since TOPP-RA does not require explicit computation of the switch points (they are implicitly identified by computing the controllable sets) the algorithm avoids one of the major implementation difficulties of TOPP-NI;
- TOPP-RA requires additional post-processing to remove jitters. See the footnote in Section 3.5.2 for more details.

3.8.2. Proof of optimality (case with no zero-inertia point)

The optimality of TOPP-RA relies on the properties of the maximal transition functions.

Definition 8. At a given stage i, the minimal and maximal controls at state x are

defined by [3]

$$\alpha_i(x) := \min\{u \mid \bar{\mathbf{a}}_i u + \bar{\mathbf{b}}_i x + \bar{\mathbf{c}}_i \leq 0\},$$
$$\beta_i(x) := \max\{u \mid \bar{\mathbf{a}}_i u + \bar{\mathbf{b}}_i x + \bar{\mathbf{c}}_i \leq 0\}.$$

The *minimal and maximal transition functions* are defined by

$$T_i^{\alpha}(x) := x + 2\Delta\alpha_i(x), \quad T_i^{\beta}(x) := x + 2\Delta\beta_i(x). \qquad \Diamond$$

The key observation is: if the maximal transition function is *non-decreasing*, then the greedy strategy of TOPP-RA is optimal. This is made precise by the following lemma.

Lemma 1. *If for all i, the maximal transition function is non-decreasing, i.e.,*

$$\forall x, x'' \in \mathcal{K}_i, \quad x \geq x' \implies T_i^{\beta}(x) \geq T_i^{\beta}(x'),$$

then TOPP-RA produces the optimal parameterization.

Proof. Consider an arbitrary admissible parameterization $\dot{s}_0^2 = x_0, u_0, \ldots, u_{N-1}, x_N = \dot{s}_N^2$. We show by induction that, for all $i = 0, \ldots, N$, $x_i^* \geq x_i$, where the sequence (x_i^*) denotes the parameterization returned by TOPP-RA (Algorithm 1).

Initialization: One has $x_0 = \dot{s}_0^2 = x_0^*$, so the assertion is true at $i = 0$.

Induction: Steps 8 and 9 of Algorithm 1 can in fact be rewritten as follows

$$x_{i+1}^* := \min\{T_i^{\beta}(x_i^*), \max(\mathcal{K}_{i+1})\}.$$

By the induction hypothesis, one has $x_i^* \geq x_i$. Since $x_i^*, x_i \in \mathcal{K}_i$, one has

$$T_i^{\beta}(x_i^*) \geq T_i^{\beta}(x_i) \geq x_{i+1}.$$

Thus,

$$\min\{T_i^{\beta}(x_i^*), \max(\mathcal{K}_{i+1})\} \geq \min\{x_{i+1}, \max(\mathcal{K}_{i+1})\}, \text{ i.e.}$$
$$x_{i+1}^* \geq x_{i+1}.$$

We have shown that at every stage the parameterization $x_0^*, \ldots, x_N^*$ has a higher velocity than that of any admissible parameterization, hence it is optimal. □

Unfortunately, the maximal transition function is not always non-decreasing, as made clear by the following lemma.

[3] These definitions differ from the common definitions of maximal and minimal controls. See Appendix A for more details.

Lemma 2. *Consider a stage i, there exists x_i^β such that, for all $x, x' \in \mathcal{K}_i$*

$$x \leq x' \leq x_i^\beta \implies T_i^\beta(x) \leq T_i^\beta(x'),$$
$$x_i^\beta \leq x \leq x' \implies T_i^\beta(x) \geq T_i^\beta(x').$$

In other words, T_i^β is non-decreasing below x_i^β and is non-increasing above x_i^β.

Similarly, there exists x_i^α such that for all $x, x' \in \mathcal{K}_i$

$$x_i^\alpha \leq x \leq x' \implies T_i^\alpha(x) \leq T_i^\alpha(x'),$$
$$x \leq x' \leq x_i^\alpha \implies T_i^\alpha(x) \geq T_i^\alpha(x').$$

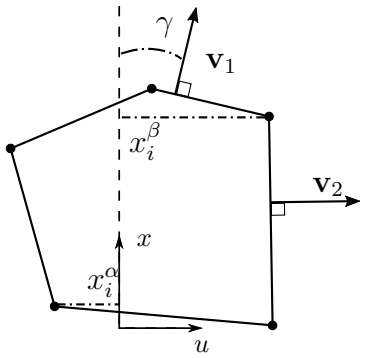


Figure 3.8.: At any stage, the polygon of controllable states and controls $\Omega_i \cap (\mathbb{R} \times \mathcal{K}_i)$ contains x_i^β: the highest state under which the transition function T_i^β is non-decreasing and x_i^α: the lowest state above which the transition function T_i^α is non-decreasing.

Proof. Consider a state x. In the (u, x) plane, draw a horizontal line at height x. This line intersects the polygon $\Omega_i \cap (\mathbb{R} \times \mathcal{K}_i)$ at the minimal and maximal controls. See Fig 3.8.

Consider now the polygon $\Omega_i \cap (\mathbb{R} \times \mathcal{K}_i)$. Suppose that one enumerates the edges counter-clockwise (ccw), then the normals of the enumerated edges also rotate ccw. For example in Fig. 3.8, the normal v_1 of edge 1 can be obtained by rotating ccw the normal v_2 of edge 2.

Let γ denote the angle between the vertical axis and the normal vector of the active constraint k at $(x, \beta(x))$. One has $\cot \gamma = \bar{\mathbf{b}}_i[k]/\bar{\mathbf{a}}_i[k]$.

As x increases, γ decreases in the interval $(\pi, 0)$. Let x_i^β be the lowest x such that, for all $x > x_i^\beta$, $\gamma < \cot^{-1}(1/(2\Delta))$ ($x_i^\beta := \max \mathcal{K}_i$ if there is no such x).

Consider now a $x > x_i^\beta$, one has, by construction

$$\frac{\bar{\mathbf{b}}_i[k]}{\bar{\mathbf{a}}_i[k]} > \frac{1}{2\Delta}, \tag{3.12}$$

where k is the active constraint at $(x, \beta_i(x))$. The maximal transition function can be written as

$$\begin{aligned} T_i^\beta(x) &= x + 2\Delta\beta_i(x) \\ &= x + 2\Delta\frac{-\bar{\mathbf{c}}_i[k] - \bar{\mathbf{b}}_i[k]x}{\bar{\mathbf{a}}_i[k]} \\ &= x\left(1 - 2\Delta\frac{\bar{\mathbf{b}}_i[k]}{\bar{\mathbf{a}}_i[k]}\right) - 2\Delta\frac{\bar{\mathbf{c}}_i[k]}{\bar{\mathbf{a}}_i[k]}. \end{aligned} \tag{3.13}$$

Since the coefficient of x is negative, $T_i^\beta(x)$ is non-increasing. Similarly, for $x \leq x_i^\beta$, $T_i^\beta(x)$ is non-decreasing. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. As there is no zero-inertia point, by uniform continuity, the $\bar{\mathbf{a}}(s)[k]$ are bounded away from 0. We can thus chose a step size Δ_{thr} such that

$$\frac{1}{2\Delta_{\text{thr}}} > \max_{s,k}\left\{\frac{\bar{\mathbf{b}}(s)[k]}{\bar{\mathbf{a}}(s)[k]} \mid \bar{\mathbf{a}}(s)[k] > 0\right\}.$$

For any step size $\Delta < \Delta_{\text{thr}}$, there is by construction no constraint that can have an angle $\gamma < \cot^{-1}(1/(2\Delta))$. Thus, for all stages i, one has $x_i^\beta = \max \mathcal{K}_i$, or in other words, that $T_i^\beta(x)$ is non-decreasing in the whole set $\mathcal{K}_i$. By Lemma 1, TOPP-RA returns the optimal parameterization. □

3.8.3. Proof of asymptotic optimality (case with zero-inertia points)

In the presence of a zero-inertia point, one cannot bound the $\bar{\mathbf{a}}(s)[k]$ away from zero. Therefore, for any step size Δ, there is an interval around the zero-inertia point where the maximal transition function is *not* monotonic over the whole controllable set $\mathcal{K}_i$. Our strategy is to show that the sub-optimality gap caused by that interval decreases to 0 with Δ.

We first identify a "perturbation interval". For simplicity, assume that the zero-inertia point $s^\bullet$ is exactly at $s_{i^\bullet}$.

Lemma 3. *There exists an integer l such that, for small enough Δ, the maximal transition function is non-decreasing at all stages except in $[i^\bullet + 1, \dots, i^\bullet + l]$.*

Proof. Consider the Taylor expansion around $s^\bullet$ of the constraint that triggers the zero-inertia point

$$\begin{aligned} \bar{\mathbf{a}}(s)[k] &= A'(s - s^\bullet) + o(s - s^\bullet), \\ \bar{\mathbf{b}}(s)[k] &= B + B'(s - s^\bullet) + o(s - s^\bullet). \end{aligned}$$

Without loss of generality, suppose $A' > 0$. Eq. (3.12) can be written for stage $i^\bullet + r$ as follows

$$2\Delta(B + B'r\Delta) > A'r\Delta + o(r\Delta).$$

Thus, in the limit $\Delta \to 0$, for $r > l := \mathrm{ceil}(2B/A')$, Eq. (3.12) will not be fulfilled by constraint k. Using the construction of Δ_{thr} in the proof of Theorem 2, one can next rule out all the other constraints at all stages. □

We now construct a "perturbation strip" by defining an upper and an lower boundaries. See Fig. 3.9 for an illustration.

Definition 9. Define states $(\kappa_i)_{i\in[i^\bullet+1,i^\bullet+l+1]}$ by

$$\begin{aligned}\kappa_{i^\bullet+1} &:= x^\beta_{i^\bullet+1},\\ \kappa_i &:= \min({T_{i-1}}^\beta(\kappa_{i-1}), x^\beta_i),\ i = i^\bullet+2,\dots,i^\bullet+l+1.\end{aligned}$$

Next, define $(\lambda_i)_{i\in[i^\bullet+1,i^\bullet+l+1]}$ by

$$\begin{aligned}\lambda_{i^\bullet+1} &= \kappa_{i^\bullet+1},\\ \lambda_i &= \min(T^\alpha_{i-1}(\kappa_{i-1}), x^\beta_i),\ i = i^\bullet+2,\dots,i^\bullet+l+1.\end{aligned}$$

Finally, define $(\mu_i)_{i\in[i^\bullet+1,i^\bullet+l+1]}$ as the highest profile that can be obtained by repeated applications of T^β and that remains below the (λ_i). ◊

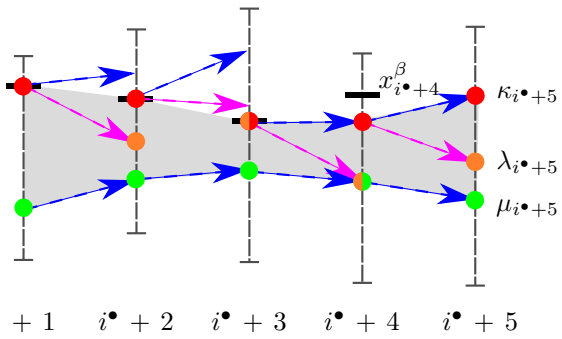


Figure 3.9.: The "perturbation strip" contains three vertical boundaries: (κ_i) [red dots], (λ_i) [orange dots] and (μ_i) [green dots]. The states (x^β_i) [thick horizontal black lines] and the controllable sets $(\mathcal{K}_i)$ [vertical intervals] are both shown.

The (κ_i) and (μ_i) form respectively the upper and the lower boundaries of the "perturbation strip". Before going further, let us establish some estimates on the size of the strip.

Lemma 4. *There exist constants C_κ and C_μ such that, for all $i \in [i^\bullet + 1, i^\bullet + l + 1]$,*

$$\max \mathcal{K}_{i^\bullet+1} - \kappa_i \leq l C_\kappa \Delta, \tag{3.14}$$

$$\max \mathcal{K}_{i^\bullet+1} - \mu_i \leq l C_\mu \Delta. \tag{3.15}$$

Proof. Let C be the upper-bound of the absolute values of all admissible controls α, β over whole segment. One has

$$\mathcal{K}_{i^\bullet+1} - \kappa_{i^\bullet+1} = \mathcal{K}_{i^\bullet+1} - x^\beta_{i^\bullet+1} \leq$$

$$(\beta_i(x^\beta_{i^\bullet+1}) - \alpha_i(x^\beta_{i^\bullet+1})) \tan(\gamma) \leq 2C\Delta.$$

Next, by definition of κ, one can see that the difference between two consecutive κ_i, κ_{i+1} is bounded by $2C\Delta$. This shows Eq. (3.14).

Since (μ_i) is the highest profile below (λ_i), there exists one index p such that $\mu_p = \lambda_p$. Thus, $\kappa_p - \mu_p = \kappa_p - \lambda_p \leq 2C\Delta$, where the last inequality comes from the definition of λ. Remark finally that the difference between two consecutive μ_i, μ_{i+1} is also bounded by $2C\Delta$. This shows Eq. (3.15). □

We now establish two properties of the "perturbation strip".

Lemma 5 (and definition)**.** *Let $J^*_i(x)$ denote TOPP-RA's cost-to-go: the cost of the profile produced by TOPP-RA starting from x at the i-stage, and $J^\dagger_i(x)$ the optimal cost-to-go.*

(a) In the interval $[\min \mathcal{K}_i, \mu_i]$, $J^*_i(x)$ equals $J^\dagger_i(x)$ and is non-increasing;

(b) For all $i \in [i^\bullet + 1, \dots, i^\bullet + l]$,

$$x \in [\mu_i, \kappa_i] \implies T^\dagger_i(x), T^\beta_i(x) \in [\mu_{i+1}, \kappa_{i+1}], \tag{3.16}$$

where $T^\dagger_i(x)$ is the optimal transition.

Proof. (a) We use backward induction from $i^\bullet + l$ to $i^\bullet + 1$.

Initialization: One has $x \leq \mu_{i^\bullet+l} \leq \lambda_{i^\bullet+l} \leq x^\beta_{i^\bullet+l}$. It follows that $T^\beta_{i^\bullet+l}(x)$ is non-decreasing over the interval $[\min \mathcal{K}_{i^\bullet+l}, \mu_{i^\bullet+l}]$.

As there is no constraint verifying Eq. (3.12) at stages $i = i^\bullet + l + 1, \dots, N$, the cost-to-go $J^*_{i^\bullet+l+1}(x)$ is non-increasing and equals the optimal cost-to-go $J^\dagger_{i^\bullet+l+1}(x)$ by Theorem 2. Choosing the greedy control at the $i^\bullet + l$-stage is therefore optimal. Next, note that

$$J^*_{i^\bullet+l}(x) = \frac{\Delta}{\sqrt{x}} + J^*_{i^\bullet+l+1}(T^\beta_{i^\bullet+l}(x)), \tag{3.17}$$

since $J^*_{i^\bullet+l+1}(x)$ and $T^\beta_{i^\bullet+l}(x)$ are non-increasing and non-decreasing respectively over $[\min \mathcal{K}_{i^\bullet+l+1}, \mu_{i^\bullet+l+1}]$ and $[\min \mathcal{K}_{i^\bullet+l}, \mu_{i^\bullet+l}]$, it follows that $J^*_{i^\bullet+l}(x)$ is non-increasing over the interval $[\min \mathcal{K}_{i^\bullet+l}, \mu_{i^\bullet+l}]$.

Induction: Suppose the hypothesis is true for $i+1 \in \{i^\bullet+2, \dots, i^\bullet+l\}$. Since $x \le \mu_i \le \lambda_i \le x_i^\beta$, one has that $T_i^\beta(x) \le \mu_{i+1}$ and that $T_i^\beta(x)$ is non-decreasing over the interval $[\min \mathcal{K}_i, \mu_i]$. Note that

$$J_i^*(x) = \frac{\Delta}{\sqrt{x}} + J^*_{i+1}(T_i^\beta(x)). \tag{3.18}$$

By the induction hypothesis, $J^*_{i+1}(T_i^\beta(x))$ is non-increasing, it then follows that $J_i^*(x)$ is non-decreasing and that $\beta_i(x)$ is the optimal control and thus $J_i^*(x) = J_i^\dagger(x)$.

(b) The part that $T_i^\dagger(x), T_i^\beta(x) \le \kappa_{i+1}$ is clear from the definition of κ. We first show $\mu_{i+1} \le T_i^\beta(x)$.

Suppose first $x \le x_i^\beta$. Then T_i^β is non-decreasing in $[\mu_i, x]$, which implies $T_i^\beta(x) \ge T_i^\beta(\mu_i) = \mu_{i+1}$.

Suppose now $x \ge x_i^\beta$. One can choose a step size Δ such that $x_i^\alpha < x_i^\beta$, which implies $x > x_i^\alpha$. One then has $T_i^\beta(x) \ge T_i^\alpha(x) \ge T_i^\alpha(x_i^\beta) \ge \lambda_{i+1} \ge \mu_{i+1}$.

Finally, to show that $\mu_{i+1} \le T_i^\dagger(x)$, we reason by contradiction. Suppose $T_i^\dagger(x) < \mu_{i+1}$. By (a), $J^\dagger_{i+1}$ is non-increasing below μ_{i+1}, thus $J^\dagger_{i+1}(T_i^\dagger(x)) > J^\dagger_{i+1}(\mu_{i+1})$ (*). On the other hand, since $T_i^\dagger(x) < \mu_{i+1} \le T_i^\beta(x)$, there exists an admissible control that steers x towards μ_{i+1}. Since $T_i^\dagger$ is the true optimal transition from x, $J^\dagger_{i+1}(\mu_{i+1}) \ge J^\dagger_{i+1}(T_i^\dagger(x))$. This contradicts (*). □

We are now ready to prove Theorem 3.

Proof of Theorem 3. Recall that $(\dot{s}_0^2 = x_0^*, \dots, x_N^*)$ is the profile returned by TOPP-RA and $(\dot{s}_0^2 = x_0^\dagger, \dots, x_N^\dagger)$ is the true optimal profile. By definition of the time-optimal cost functions, we can expand the initial costs $J_0^*(\dot{s}_0^2)$ and $J_0^\dagger(\dot{s}_0^2)$ into three terms as follows

$$J_0^*(\dot{s}_0^2) = \sum_{i=0}^{i^\bullet} \frac{\Delta}{\sqrt{x_i^*}} + \sum_{i=i^\bullet+1}^{i^\bullet+l} \frac{\Delta}{\sqrt{x_i^*}} + J^*_{i^\bullet+l+1}(x^*_{i^\bullet+l+1}), \tag{3.19}$$

and

$$J_0^\dagger(\dot{s}_0^2) = \sum_{i=0}^{i^\bullet} \frac{\Delta}{\sqrt{x_i^\dagger}} + \sum_{i=i^\bullet+1}^{i^\bullet+l} \frac{\Delta}{\sqrt{x_i^\dagger}} + J^\dagger_{i^\bullet+l+1}(x^\dagger_{i^\bullet+l+1}), \tag{3.20}$$

(a) Applying Theorem 2, for small enough Δ, one can show that

$$\forall i \in [0, \dots, i^\bullet+1],\ x_i^* \ge x_i^\dagger. \tag{3.21}$$

Thus, the first term of $J_0^*(\dot{s}_0^2)$ is smaller than the first term of $J_0^\dagger(\dot{s}_0^2)$.

(b) Suppose $x^*_{i^\bullet+1}, x^\dagger_{i^\bullet+1} \in [\mu_{i^\bullet+1}, \kappa_{i^\bullet+1}]$. From Lemma 5(b), one has for all $i \in [i^\bullet+1, \dots, i^\bullet+l]$, $x^*_i, x^\dagger_i \in [\mu_i, \kappa_i]$. Thus, using the estimates of Lemma 4, the second terms can be bounded as follows

$$\sum_{i=i^\bullet+1}^{i^\bullet+l} \frac{\Delta}{\sqrt{x^*_i}} \leq \frac{l\Delta}{\sqrt{\max \mathcal{K}_{i^\bullet+1} - C_\mu \Delta l}}, \text{ and}$$

$$\sum_{i=i^\bullet+1}^{i^\bullet+l} \frac{\Delta}{\sqrt{x^\dagger_i}} \geq \frac{l\Delta}{\sqrt{\max \mathcal{K}_{i^\bullet+1} + C_\kappa \Delta l}}.$$

Thus

$$\sum_{i=i^\bullet+1}^{i^\bullet+l} \frac{\Delta}{\sqrt{x^*_i}} - \sum_{i=i^\bullet+1}^{i^\bullet+l} \frac{\Delta}{\sqrt{x^\dagger_i}} \leq \frac{(C_\mu + C_\kappa)\Delta^2 l^2}{2\sqrt{\max \mathcal{K}_{i^\bullet+1} - C_\mu \Delta l}}.$$

If $x^*_{i^\bullet+1}, x^\dagger_{i^\bullet+1} < \mu_{i^\bullet+1}$, by Lemma 5(a) it is easy to see that $J^*_0(\dot{s}^2_0) = J^\dagger_0(\dot{s}^2_0)$.

If $x^*_{i^\bullet+1} \geq \mu_{i^\bullet+1} > x^\dagger_{i^\bullet+1}$, then by Lemma 5, $x^*_{i^\bullet+l+1} \geq \mu_{i^\bullet+l+1} > x^\dagger_{i^\bullet+l+1}$, which implies next that $J^*_0(\dot{s}^2_0) < J^\dagger_0(\dot{s}^2_0)$, which is impossible.

(c) Regarding the third terms, observe that, by applying Theorem 2 over $[i^\bullet + l + 1, \dots, N]$, one has $J^*_{i^\bullet+l+1}(x) = J^\dagger_{i^\bullet+l+1}(x)$ for all $x \in \mathcal{K}_{i^\bullet+l+1}$. Thus

$$J^*_{i^\bullet+l+1}(x^*_{i^\bullet+l+1}) - J^\dagger_{i^\bullet+l+1}(x^\dagger_{i^\bullet+l+1}) =$$

$$J^\dagger_{i^\bullet+l+1}(x^*_{i^\bullet+l+1}) - J^\dagger_{i^\bullet+l+1}(x^\dagger_{i^\bullet+l+1}) \leq$$

$$C_{J^\dagger} |x^*_{i^\bullet+l+1} - x^\dagger_{i^\bullet+l+1}| \leq C_{J^\dagger}(C_\mu + C_\kappa)\Delta l,$$

where $C_{J^\dagger}$ is the Lipshitz constant of $J^\dagger$.

Grouping together the three estimates (a), (b), (c) leads to the conclusion of the theorem. □

3.8.4. Error analysis of discretization schemes

First-order interpolation scheme

In the main text the collocation discretization scheme was presented to discretize the constraints over $N+1$ grid points. We now introduce another scheme: first-order interpolation scheme.

In this scheme, at stage i, we require (u_i, x_i) and $(u_i, x_i + 2\Delta_i u_i)$ to satisfy the constraints at $s = s_i$ and $s = s_{i+1}$ respectively. That is, for $i = 0, \dots, N-1$

$$\begin{bmatrix} \mathbf{a}(s_i) \\ \mathbf{a}(s_{i+1}) + 2\Delta \mathbf{b}(s_{i+1}) \end{bmatrix} u + \begin{bmatrix} \mathbf{b}(s_i) \\ \mathbf{b}(s_{i+1}) \end{bmatrix} x + \begin{bmatrix} \mathbf{c}(s_i) \\ \mathbf{c}(s_{i+1}) \end{bmatrix} \in \begin{bmatrix} \mathcal{C}(s_i) \\ \mathcal{C}(s_{i+1}) \end{bmatrix} \tag{3.22}$$

At $i = N$, one uses only the top half of the above equations. By appropriately rearranging the terms, the above equations can finally be rewritten as

$$\mathbf{a}_i u + \mathbf{b}_i x + \mathbf{c}_i \in \mathcal{C}_i. \tag{3.23}$$

Error analysis

For simplicity, suppose Assumption 1 holds. That is, there exists $\tilde{\mathbf{a}}(s)_{s\in[0,1]}, \tilde{\mathbf{b}}(s)_{s\in[0,1]}, \tilde{\mathbf{c}}(s)_{s\in[0,1]}$ which define the set of admissible control-state pairs. Consider the interval $[s_0, s_1]$, the parameterization is given by

$$x(s; u_0, x_0) = x_0 + 2su_0,$$

where x_0 is the state at s_0 and u_0 is the constant control along the interval. Note that $s_0 = 0, s_1 = \Delta$.

Define the constraint satisfaction *function* by

$$\epsilon(s) := u_0\tilde{\mathbf{a}}(s) + x(s)\tilde{\mathbf{b}}(s) + \tilde{\mathbf{c}}(s). \tag{3.24}$$

The *greatest* constraint satisfaction error over $[s_0, s_1]$ is

$$\max\left\{\max_{k,s\in[s_0,s_1]} \epsilon(s)[k], 0\right\}.$$

Different discretization schemes enforce different conditions on $\epsilon(s)$. In particular, one has for

- *collocation scheme*: $\epsilon(s_0) \leq 0$;
- *first-order interpolation scheme*: $\epsilon(s_0) \leq 0$, $\epsilon(s_1) \leq 0$.

Using the classic result on Error of Polynomial Interpolation [122, Theorem 2.1.4.1], one obtains the following estimate of $\epsilon(s)$ for the *collocation scheme*:

$$\epsilon(s) = \epsilon(s_0) + (s - s_0)\epsilon'(\xi) = s\epsilon'(\xi),$$

for $\xi \in [s_0, s]$. Suppose the derivatives of $\tilde{\mathbf{a}}(s), \tilde{\mathbf{b}}(s), \tilde{\mathbf{c}}(s)$ are bounded, one has

$$\max_{s\in[s_0,s_1]} \epsilon(s) = O(\Delta).$$

Thus the greatest constraint satisfaction error of the collocation discretization scheme has order $O(\Delta)$.

Using the same theorem, one obtains the following estimation of $\epsilon(s)$ for the *first-order interpolation scheme*:

$$\begin{aligned}\epsilon(s) =& \epsilon(s_0) + (s - s_0)\frac{\epsilon(s_1) - \epsilon(s_0)}{s_1 - s_0} + \frac{(s - s_0)(s - s_1)\epsilon''(\xi)}{2!} \\ =& \frac{s(s - \Delta)\epsilon''(\xi)}{2!},\end{aligned}$$

for some $\xi \in [s_0, s_1]$. Again, since the derivatives of $\tilde{\mathbf{a}}(s), \tilde{\mathbf{b}}(s), \tilde{\mathbf{c}}(s)$ are assumed to be bounded, one has

$$\max_{s \in [s_0, s_1]} \epsilon(s) = O(\Delta^2).$$

Thus the greatest constraint satisfaction error of the first-order interpolation discretization scheme has order $O(\Delta^2)$.

Chapter 4.

Critically fast pick-and-place with suction cups

4.1. Introduction

Suction cups have proved to be some of the most robust and versatile devices to grasp a variety of objects, and are therefore used in a large proportion of automation solutions in logistics (factory lines, e-commerce, etc.) In addition to robustness and versatility, cycle time is another key driver in the development of automation: how fast automated systems can pick and place objects is a decisive economic question.

In this chapter, we are interested in planning and executing critically fast pick-and-place movements with suction cups. By "critically fast" we mean the fastest possible movements for transporting an object such that it does not slip or fall from the suction cup. Another major concern is to also minimize the planning time. Indeed, in a majority of e-commerce scenarios, the robot movements must be computed based on the dynamically perceived positions of the objects to be transported, which implies that both planning and execution times contribute to the total cycle time.

Note that, to focus on the planning and execution aspects, we assume in this chapter that the perception problem has been solved upstream: (i) a good model of the environment is available, (ii) the geometries, weight distribution, and initial positions/orientations of the objects are given and accurate.

Essentially, ensuring that the object being transported does not slip or fall amounts to guaranteeing that, at every moment, the net inertial wrench acting on the object can be physically "realized" by its contact with the suction cup (weak contact stability [93, 25]). Since inertial wrenches can be easily computed given a sufficiently accurate model of the object, this task is reduced to identifying the set of wrenches

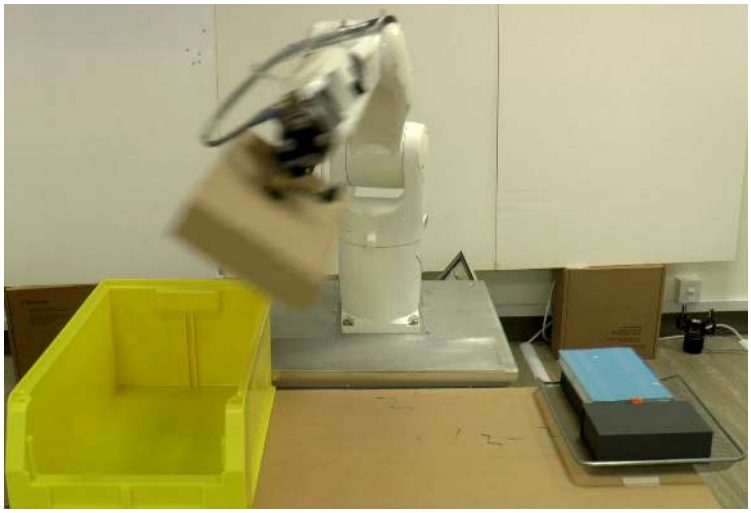


Figure 4.1.: Critically fast pick-and-place of a box with a suction cup. The full video of the experiment is available at https://youtu.be/b9H-zOYWLbY.

that are *physically realizable* for the given suction cup contact. A procedure for doing this is among the contributions of this chapter. In the following, we will refer to this objective as satisfying the *suction cup grasp stability constraint*, or simply *grasp stability constraint*.

In industrial robotic pick-and-place systems, a popular approach to maintaining suction cup grasp stability is to uniformly restrict the robot's joint velocity and acceleration. The velocity and acceleration limits can be tuned by executing multiple test trajectories, then selecting the set of limits at which there is no failure and the average duration is the shortest. A drawback of this approach is that testing must be done for each object/suction cup combination, and therefore, is tedious and time consuming. Furthermore, it is certain that the resulting movements are not optimal, since the kinematic limits are chosen with respect to *all* testing trajectories. Also, there is no guarantee that the robot can successfully execute any new, untested trajectory during its actual operation.

An equally important concern is that the computational cost of planning should be small, as otherwise it would undermine the benefit of controlling the robot to move critically fast. For example, suppose one is capable of computing the true time-optimal trajectories, but requires a few seconds per trajectory. In this case, it might be more reasonable to use a sub-optimal approach that has cheaper computational cost, such as limiting the robot's joint velocity and acceleration. In the same way, the planning procedure needs to be reliable: it has to detect infeasible instances correctly, and is robust against numerical errors.

In this chapter, we propose an approach for planning critically fast movements for robotic pick-and-place with suction cups. This approach is computationally efficient

and robust, and is verified experimentally to be capable of producing near time-optimal movements. This performance is achieved by means of three main technical contributions:

- A model for suction cup contacts (Section 4.2.3);
- A procedure to compute and approximate the contact stability constraint based on that model (Section 4.2.4);
- A procedure to parameterize, in a time-optimal manner, arbitrary geometric paths under the identified contact stability constraint (Section 4.3).

The full pipeline is available as open-source[1]. Experimental results are reported and discussed in Section 4.4.

4.2. Grasp Stability for suction cups

4.2.1. Background 1: Linearized friction cone

In suction cup grasping, frictional forces exist between the suction cup's pad and the object; these forces are fundamental for maintaining a stable grasp. These frictional forces are modelled using the Colomb friction model. Let $\mathbf{f} = (f_x, f_y, f_z)$ denote a friction force vector, μ denote the coefficient of friction and let the Z-axis be the normal contact direction, the Colomb friction model states

$$\|(f_x, f_y)\|_2 \leq \mu f_z. \tag{4.1}$$

The set of feasible friction forces, or equivalently the feasible set of inequality (6.20), is a Second-Order cone[2] [20] and hence, can be approximated by a polyhedral cone with arbitrary precision [7]. For instance, an approximation with 4 linear inequalities is given as follows:

$$\begin{bmatrix} -1 & -1 & -\mu \\ -1 & 1 & -\mu \\ 1 & 1 & -\mu \\ 1 & -1 & -\mu \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \leq \mathbf{0}_4. \tag{4.2}$$

This approximation is known as the *linearized friction cone*. This is also the approximation we use in this chapter.

[1] https://github.com/hungpham2511/rapid-transport
[2] Also known as a Lorentz cone or an ice-cream cone.

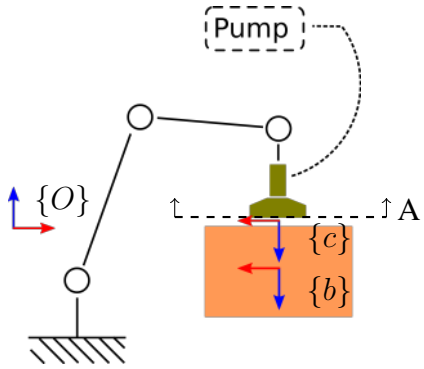


Figure 4.2.: Diagram of a robotic system for pick-and-place.

4.2.2. Background 2: Polyhedral computations

A polytope can be defined as the feasible set of a system of linear equalities and inequalities:

$$\mathbf{Ap} \leq \mathbf{b},\ \mathbf{Cp} = \mathbf{d},$$

where $\mathbf{A}, \mathbf{b}, \mathbf{C}, \mathbf{d}$ are matrices of suitable dimensions. This representation is known as the H-representation, where H stands for halfspace. Alternatively, a polytope can also be defined as the Minkowski sum[3] of the convex hull of a finite number of points and the conic hull of a finite number of rays:

$$\left\{\sum_i \theta_i \mathbf{p}_i \mid \sum_i \theta_i = 1; \forall i, \theta_i \geq 0\right\} + \left\{\sum_i \theta_i \mathbf{l}_i \mid \forall i, \theta_i \geq 0\right\}.$$

The vertices $\mathbf{p}_i$ (resp. rays $\mathbf{l}_i$) are referred to as the *generating* vertices (resp. rays). This representation is known as the V-representation, where V stands for vertex.

A seminal result in the theory of polyhedral computation, discovered by Minkowski and Weyl, is that any polytope has both a H-representation and a V-representation [44]. The problem of computing the V-representation given the H-representation is *vertex enumeration*; and the dual problem is *facet enumeration*. Even though both problems are known to be NP-hard, algorithms, such as the Double Description method [44] or the Reversed Search algorithm [3], have been developed to solve them in reasonable times. Both algorithms have publicly available implementations.

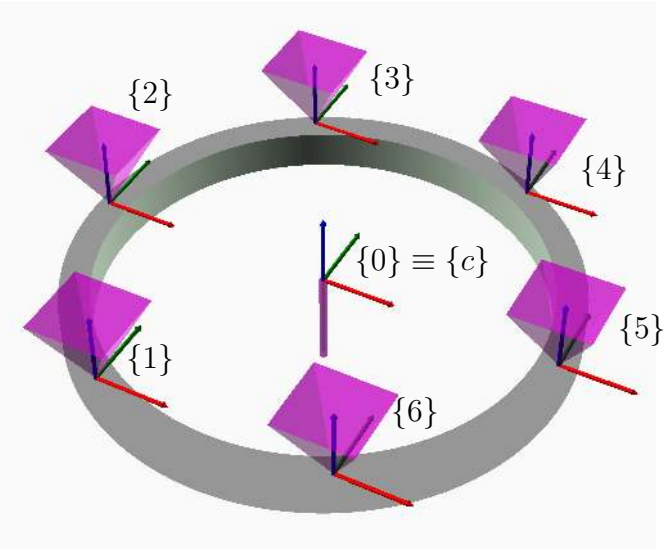


Figure 4.3.: Cross-sectional view (cross section A in Fig. 4.2) of the contact between the cup and the object. The physical contact area is modelled by m point forces following the linearized Colomb friction model and 1 point force capturing the suction force due to negative pressure. A local frame is defined at each point contact.

4.2.3. A model for suction cup grasping

The interface between the suction cup on the object is modelled as consisting of a collection of point forces: one suction force and m contact points [4] (See Fig. 4.3). The suction force, denoted by $\mathbf{f}^{(0)}$, satisfies

$$\mathbf{f}_0^{(0)} = [0, 0, PA]^\top, \tag{4.3}$$

where P is the negative pressure in Pa and A is the area of the suction cup in m^2. At the i-th contact point, the cup exerts on the object a force $\mathbf{f}^{(i)}$ that follows the linearized Colomb friction model (4.2). The suction and contact forces $\mathbf{f}^{(i)}, i = 0, \ldots, m$ with the corresponding local frames $\{i\}, i = 0, \ldots, m$ are depicted in Fig. 4.3.

The contact wrench exerted on the object by the cup depend on the suction and contact forces. Indeed, let $\{c\}$ denote a frame that is attached rigidly to the suction cup. By transforming all individual forces to frame $\{c\}$ and adding them together, one

[3] The Minkoski sum of two sets $\mathcal{G}, \mathcal{H}$ is defined as

$$\{x + y \mid x \in \mathcal{G}, y \in \mathcal{H}\}$$

[4] From [25], adopting the point-force formulation does not incur any loss in generality as compared to the surface-force formulation

obtains the net wrench $\mathbf{w}_c$:

$$\mathbf{w}_c = \sum_{i=0}^{m} \mathbf{G}_{ci} \mathbf{f}_i^{(i)}, \tag{4.4}$$

where the matrix $\mathbf{G}_{ci}$ is defined via the position vector $\mathbf{p}_{ci}$ and rotational matrix $\mathbf{R}_{ci}$ of frame $\{i\}$ in frame $\{c\}$ as below

$$\mathbf{G}_{ci} = \begin{bmatrix} [\mathbf{p}_{ci}\times]\mathbf{R}_{ci} \\ \mathbf{R}_{ci} \end{bmatrix}.$$

We make an assumption concerning the realizability of a contact wrench: a contact wrench $\mathbf{w}_c$ is physically realizable if there exists a set of individual contact forces $\mathbf{f}^{(0)}, \ldots, \mathbf{f}^{(m)}$ that satisfies Equation (4.2) and (4.3). This condition is often used in humanoid locomotion [26] and is known as the *weak contact stability* condition [93].

The *exact* grasp stability constraint can now be shown to have the form

$$\mathbf{F}_c \mathbf{w}_c \leq \mathbf{g}_c, \tag{4.5}$$

where $\mathbf{F}_c$ and $\mathbf{g}_c$ are matrices of appropriate sizes. We demonstrate this result by presenting the below constructive procedure:

1. let $\hat{\mathcal{F}}$ be the set of feasible values of $\hat{\mathbf{f}} := \{\mathbf{f}^{(0)}, \ldots, \mathbf{f}^{(m)}\}$; form the H-representation of $\hat{\mathcal{F}}$;

2. compute the corresponding V-representation, which is a set of generating vertices[5] $(\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_l)$;

3. use Equation (4.4) to transform $(\hat{\mathbf{f}}_1, \ldots, \hat{\mathbf{f}}_l)$ to $(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_l)$; $\mathcal{W}_c$ is the convex hull of $(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_l)$;

4. compute the H-representation of $\mathcal{W}_c$, which are the coefficient matrices $\mathbf{F}_c, \mathbf{g}_c$ in Equation (4.5).

Refer to Section 4.2.2 for a discussion on transforming between the two representations (H and V) of a polytope. A more detail application of this technique applied in the context of legged locomotion can be found in [24].

4.2.4. Approximating the grasp stability constraint

In practice, $\mathbf{F}_c$ might have many rows, causing computational difficulties during motion planning. We alleviate this issue by approximating the set of physically realizable wrenches as the convex hull of *a set* of points $\mathcal{Y}$ that (i) is a subset of $\mathcal{W}_c$ and (ii)

[5] There can not be generating rays since $\hat{\mathcal{F}}$ is bounded.

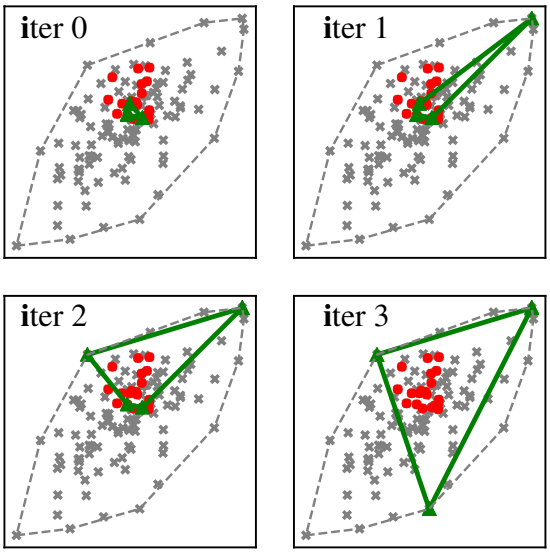


Figure 4.4.: Application of the proposed procedure for approximating grasp stability constraint on a planar data set. Vertices $\hat{\mathbf{w}}_i$ are shown as gray crosses; guiding samples are shown as red dots. The proposed procedure finds 3 vertices (iter 3) whose convex hull covers the guiding samples completely. In comparison, the dashed grey polygon, representing the exact constraint, has four times as many edges.

contains interacting wrenches that are *the most likely to be realized during execution*. To achieve the latter objective, we randomly sample potentially interacting wrenches, referred to as *guiding samples*. This can be done offline by generating a few random trajectories in the same workspace, sample a number of points in each trajectory, and for each point compute the interacting wrench using the Newton-Euler equations. Subsequently, we extend $\mathcal{Y}$ from a simple initial guess to cover more guiding samples in an interative fashion.

Concretely, the procedure proceeds as follows:

1. randomly sample N_w guiding samples; only samples that belongs to $\mathcal{W}_c$ are retained;

2. initialize $\mathcal{Y}$ as a simplex with 7 randomly chosen guiding samples as vertices;

3. compute the convex hull $\mathcal{H}$ of $\mathcal{Y}$;

4. choose the face h^* of $\mathcal{H}$ that contains the most guiding samples in its infeasible halfspace;

5. add to $\mathcal{Y}$ the vertex in $\mathcal{W}_c$ that is in the infeasible halfspace of h^* and is furthest away from it;

6. repeat from step 3 until the number of vertices in $\mathcal{Y}$ is greater than a specified value.

Remark that in step 2 since the wrench space is 6 dimensional, a simplex in this space has 7 vertices. Hence, this step amounts to choosing randomly 7 guiding samples. After each iteration, a point in $(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_l) \setminus \mathcal{Y}$ is added to $\mathcal{Y}$. We found that after about 50 iteration, the convex hull of $\mathcal{Y}$ covers a significant portion of the guiding samples and contains approximately 3000-4000 faces. In contrast, the original H-representation can have up to 150000 faces.

An application of this procedure on a planar data set is demonstrated in Fig. 4.4. There, the green triangle representing the approximate grasp constraint that has only 3 edges but cover all guiding samples. This result is computed in 3 extending iterations. In comparison, the dashed grey polygon, representing the exact constraint, has four times as many edges.

4.3. Planning critically fast movements

4.3.1. Motion planning pipeline

We implement a pipeline for planning critically fast movements following the *decoupling approach*:

1. Find a collision-free path using standard geometric planners (e.g. RRT [65]);

2. Time-parameterize the collision-free path to minimize traversal time under kinodynamic constraints (in our case: joint velocity and acceleration bounds and suction cup constraints).

Although the decoupling approach does not generally produce optimal nor even locally-optimal trajectories, it has the distinctive advantage of being robust and fast. This is due to the highly mature states of collision-free path planning and path time-parameterization techniques [101].

Regarding the latter, the problem of finding the time-optimal time-parameterization of a geometric path subject to kinodynamic constraints is a classical problem in robotics [16]. If the constraints under consideration are of First- or Second-Order (see below and also in [96]), then this problem can be solved extremely efficiently using the recently-developed Time-Optimal Path Parameterization via Reachability Analysis (TOPP-RA) algorithm [96].

In the next section, we show that the grasp stability constraint is a Second-Order constraint.

4.3.2. Reduction of the grasp stability constraint

A Second-Order constraint has the following form [96]

$$\mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}(\mathbf{q}) \in \mathscr{C}(\mathbf{q}), \tag{4.6}$$

- $\mathbf{A}, \mathbf{B}, \mathbf{f}$ are continuous mappings from $\mathbb{R}^n$ to $\mathbb{R}^{m\times n}, \mathbb{R}^{n\times m\times n}$ and $\mathbb{R}^m$ respectively;
- $\mathscr{C}(\mathbf{q})$ is a convex polytope in $\mathbb{R}^m$.

To establish that any grasp stability constraint can be reformulated as a Second-Order constraint, recall first the following relationships between the robot's joint position and the object's motion [58]:

$$\boldsymbol{\omega}_b = \mathbf{J}_{rot}(\mathbf{q})\dot{\mathbf{q}}, \qquad \boldsymbol{\alpha}_b = \mathbf{J}_{rot}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{H}_{rot}(\mathbf{q})\dot{\mathbf{q}}, \tag{4.7}$$

$$\mathbf{v}_b = \mathbf{J}_{trans}(\mathbf{q})\dot{\mathbf{q}}, \qquad \mathbf{a}_b = \mathbf{J}_{trans}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{H}_{trans}(\mathbf{q})\dot{\mathbf{q}}, \tag{4.8}$$

where $\mathbf{J}_{trans}, \mathbf{H}_{trans}, \mathbf{J}_{rot}, \mathbf{H}_{rot}$ are the translational and rotational Jacobians and Hessians, $\mathbf{v}_b, \boldsymbol{\omega}_b$ denote the object's translational and rotational velocities and $\mathbf{a}_b, \boldsymbol{\alpha}_b$ denote the translational and rotational accelerations; all are in the object's body frame $\{b\}$. Next, combining Equations (4.7) and (6.16) with the Newton-Euler equations in the body frame, which are

$$\mathbf{w}_b + \begin{bmatrix} 0_3 \\ \mathbf{g}_b m \end{bmatrix} = \begin{bmatrix} \mathbf{I}_b \boldsymbol{\alpha}_b + \boldsymbol{\omega}_b \times \mathbf{I}_b \boldsymbol{\omega}_b \\ m\mathbf{a}_b, \end{bmatrix}. \tag{4.9}$$

It follows that the interaction wrench in frame $\{b\}$ has the following form

$$\mathbf{w}_b(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \Theta_1(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \Theta_2(\mathbf{q})\dot{\mathbf{q}} + \Theta_3(\mathbf{q}),$$

where $\Theta_1, \Theta_2, \Theta_3$ are tensors depending on the geometry of the robot and the inertial properties of the object. The left hand-side shows clearly that $\mathbf{w}_b$ also depends on $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$.

Next, let $\mathbf{G}_{cb}$ be the *constant* matrix that transforms a wrench from the object's body frame $\{b\}$ to the suction cup's frame $\{c\}$. Substituting to Equation (4.5), grasp stability constraint can then be written as

$$\mathbf{F}_c \mathbf{G}_{cb} \left\{ \Theta_1(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \Theta_2(\mathbf{q})\dot{\mathbf{q}} + \Theta_3(\mathbf{q}) \right\} \leq \mathbf{g}_c. \tag{4.10}$$

It follows immediately that Equation (6.18) is a Second-Order constraint. Indeed,

$\Theta_1, \Theta_2, \Theta_3$ take the place of $\mathbf{A}, \mathbf{B}, \mathbf{f}$ respectively and the fixed convex polytope

$$\{\mathbf{w}_b \mid \mathbf{F}_c \mathbf{G}_{cb} \mathbf{w}_b \leq \mathbf{g}_c\}.$$

takes the place of $\mathscr{C}(\mathbf{q})$.

4.4. Experiments

Two aspects of the proposed approach were experimentally investigated. First, we evaluate the *quality* of the trajectories for transporting object by looking at the rate of successful transport and the trajectories' durations in 20 randomly generated instances (4.4-A). Second, we report the actual computational cost of the proposed pipeline in a realistic pick-and-place scenario (4.4-B).

4.4.1. Experimental setup

The same equipment was employed throughout the experiments. These include a position-controlled industrial robot Denso VS-060, equipped with a suction cup connected to a vacuum pump. The robot is controlled at $125\,\mathrm{Hz}$. The suction cup has a radius of $12.5\,\mathrm{mm}$ and the vacuum pump generates a negative pressure of approximately $30\,\mathrm{kPa}$. Objects considered in the experiments have varying weights ranging from $0.2\,\mathrm{kg}$ to $0.6\,\mathrm{kg}$; all objects have known weights and moment of inertia. All computations were done on a single core of a laptop running Ubuntu 16.04 at $3.800\,\mathrm{GHz}$.

Collision-free paths between two robot configurations were all computed using `OpenRAVE`'s implementation [35] of `biRRT` algorithm [65].

For time-parameterization, we used the same approximation of the grasp stability constraint in all experiments. The coefficients matrices $\mathbf{F}_c, \mathbf{g}_c$ were identified following the procedure given in Section 4.2 with the following parameters: $m = 6$ points are used to approximate the surface area contact; coefficient of friction $\mu = 0.3$ (identified in prior using a Force-Torque sensor); suction cup radius is $12.5\,\mathrm{mm}$; maximum number of vertices is 60. The resulting coefficient matrices have 3099 rows, which is significantly less than the number of rows of the exact coefficient matrices (130000 rows). All TOPP instances are discretized with 100 gridpoints, using the first-order interpolation scheme [96]. The code is written mostly in Python, with the most computationally demanding components written in Cython and C++. TOPP-RA solves the LPs using a custom LP solver based on Seidel's algorithm [108].

4.4.2. Trajectory quality

We first looked at the likelihood of the objects falling, slipping or twisting when executing the trajectories found by our pipeline. As a baseline for comparison, we also considered an alternative strategy, where one computes time-optimal time-parameterizations subject only to the robot's kinematic constraints, without taking into account the suction cup constraint.

Twenty geometric paths were randomly generated and tested. To mimic actual pick-and-place settings, we adopted the following procedure:

1. sample randomly the object's starting and goal poses, each in a dedicated region of the workspace;

2. use `OpenRAVE`'s inverse kinematics to find the corresponding robot's starting and goal configurations;

3. find a path between the starting and goal configurations using `OpenRAVE`.

In all trials, a rectangle notebook (See Fig. 4.5) with weight $0.551\,\mathrm{kg}$ and moment of inertia $\mathrm{diag}(9.28, 21.10, 29.80) \times 10^{-4}\,\mathrm{kgm}^2$ was transported with the suction cup. The contact point is $12.5\,\mathrm{mm}$ above the notebook's center of mass.

It was found that, by considering grasp stability, the object could be transported without slipping or falling for all 20 paths (Table 4.1). On the other hand, trajectories retimed without suction cup constraint had a much lower success rate: only 4 out of 20 paths could be executed successfully. Remarkably, trajectories computed considering grasp stability were not significantly slower. For example, slowing respectively trajectory number 2 by 6.3% and trajectory 14 by 20% made those trajectories safe.

A second experiment was performed to examine the degree of sub-optimality. We took paths number 12 and 18, parameterized both subject to grasp stability constraint and executed them at three levels of speed: $100\%, 120\%, 140\%$. We then inspected the pose of the object relative to the suction cup after each execution. Significant slipping of nearly $1\,\mathrm{cm}$ and $10\,\mathrm{deg}$ were found at the trials executed at 120% and 140% speed, see Fig. 4.5 and first part of the experimental video `https://youtu.be/b9H-zOYWLbY`.

4.4.3. Computational performance

To investigate the computational cost of the proposed approach, we considered a pick-and-place scenario that involves the robot picking objects using the suction cup from a bin and stacking them at a distant goal position (Fig. 4.1). The objective is to complete

Table 4.1.: Trajectory duration (in seconds) and success rate of 20 randomly generated trajectories retimed without and with suction cup constraints (SCC).

#	without SCC	with SCC	time ext. (%)
0	0.432 ✗	**0.672** ✓	35.71
1	0.568 ✗	**0.832** ✓	31.73
2	0.944 ✗	**1.008** ✓	6.35
3	0.504 ✗	**0.688** ✓	26.74
4	0.728 ✗	**0.952** ✓	23.53
5	0.504 ✗	**0.672** ✓	25.00
6	**1.064** ✓	**1.064** ✓	**0.00**
7	0.520 ✗	**0.688** ✓	24.42
8	1.136 ✗	**1.344** ✓	15.48
9	0.496 ✗	**0.744** ✓	33.33
10	**0.632** ✓	**0.728** ✓	**13.19**
11	0.520 ✗	**0.648** ✓	19.75
12	0.440 ✗	**0.728** ✓	39.56
13	**0.824** ✓	**0.896** ✓	**8.04**
14	1.104 ✗	**1.392** ✓	20.69
15	0.528 ✗	**0.664** ✓	20.48
16	0.520 ✗	**0.736** ✓	29.35
17	1.344 ✗	**1.848** ✓	27.27
18	0.472 ✗	**0.760** ✓	37.89
19	**0.512** ✓	**0.600** ✓	**14.67**

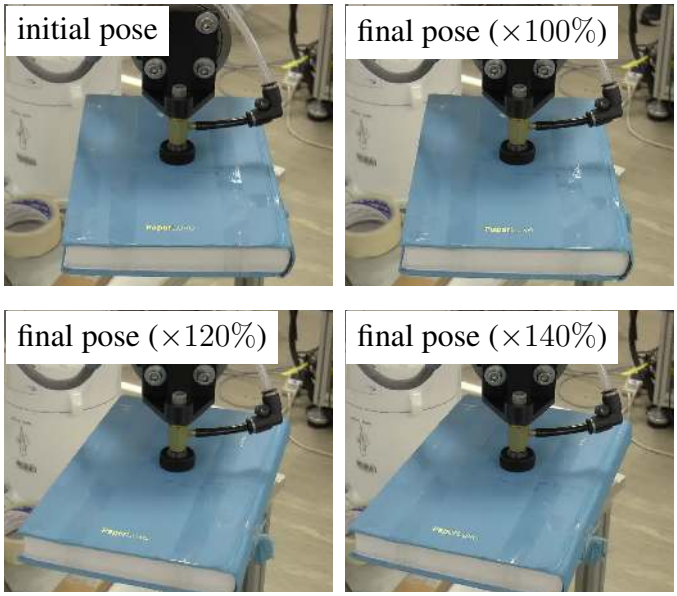


Figure 4.5.: Executing a retimed trajectory at 120% and 140% speed led to visible twisting and slipping.

Table 4.2.: Breakdown of cycle times (in seconds)

	obj 1	obj 2	obj 3	obj 4
APPROACH plan	0.018	0.053	0.016	0.041
REACH plan	0.029	0.034	0.040	0.034
MOVE plan	0.183	0.101	0.147	0.198
MOVE retime	0.111	0.111	0.124	0.137
Total plan	0.341	0.299	0.327	0.410
APPROACH exec.	0.438	0.747	0.550	0.531
REACH exec.	0.306	0.368	0.415	0.436
ATTACH delay	1	1	1	1
MOVE exec.	0.841	0.729	1.062	1.371
DETACH delay	1	1	1	1
Total exec.	3.585	3.844	4.027	4.338
Total (plan + exec.)	3.926	4.143	4.354	4.748

the task as fast as possible, planning and execution time included. The object properties, initial poses and final desired poses were determined prior to the experiment. The objects weighted between $0.204\,\mathrm{kg}$ and $0.551\,\mathrm{kg}$.

A pick-and-place cycle consisted of several phases. First, the robot *approaches* a pose that is directly on top of the bin. Then, it *reaches* for the topmost object, closes the vacuum valve, waits for 1 second for the object to *attach* to the suction cup, and finally *moves* to the given destination. When the robot arrives at the destination, it opens the valve and waits for 1 second for the object to completely *detach* before starting the next cycle. Note that the waiting times of 1 second could be reduced.

All collision-free paths were planned *online* using `OpenRAVE`. Time-parameterization was also performed *online* subject to the grasp stability constraint in addition to the robot's kinematic limits.

The experiment can be visualized in the second part of the experimental video. Table 4.2 reports our findings. Pick-and-place cycles were less than $5\,\mathrm{sec}$ for all objects. On average, the cycle time was $4.29\,\mathrm{sec}$, of which $0.34\,\mathrm{sec}$ was for trajectory planning (including path planning and time-parameterization), $1.95\,\mathrm{sec}$ was for robot motion, and $2\,\mathrm{sec}$ was for waiting for the pump.

4.5. Summary

We have proposed an approach for planning critically fast trajectories for manipulators performing pick-and-place with suction cup. Before execution, we identify the grasp stability constraint, the constraint that the object must not fall from or slip or twist

relatively to the suction cup, as a system of linear inequalities. During execution, we plan collision-free geometric paths for transporting objects and retime these paths *time-optimally* subject to the identified grasp stability constraint using the TOPP-RA algorithm.

Experiments were conducted to assess the performance of the proposed approach. The results suggest that the approach is capable of producing high-quality trajectories: these trajectories can be executed successfully without causing the object to fall or slip and have duration that were close to the true time-optimal values. Further, we also found that the proposed approach has a low computational cost, making it suitable for online motion planning for logistics applications.

There are two limitations that we are actively investigating:

- Our approach requires exact knowledge of the object's inertial properties: center-of-mass position, mass and moment of inertia and the robot's geometry. Yet, in a practical setting, identification and modelling errors always exist. Usually, only approximations of objects' properties are available. This observation leads to two questions: 1) what are the effects of identification errors on motion quality and 2) how to handle these errors;

- The approximation step presented in Section 4.2.4 reduces computational time significantly as the size of the grasp stability constraint is much smaller. However, its effects on motion quality is, in general, not completely understood.

Chapter 5.

On the Structure of Time-Optimal Path Parameterization with Third-Order Constraints

5.1. Introduction

Given a robot and a smooth path in the robot's configuration space, the problem of finding the Time-Optimal Parameterization of that Path (TOPP) with second-order constraints (e.g. bounds on acceleration, torques, contact stability) is an important and well-studied in robotics. An efficient algorithm to solve this problem was first proposed in the 1980's [16, 115] and has been continuously perfected since then, see [99] for a historical review. The algorithm has also been extended to handle a wide range of problems, from manipulators subject to torque bounds [16, 112] to vehicles or legged robots subject to balance constraints [111, 100, 100, 24], to kinodynamic motion planning [101], etc.

According to Pontryagin's Maximum Principle, time-optimal trajectories are bang-bang, which implies instantaneous switches in the second-order quantities that are constrained [16]. For instance, time-optimal trajectories with acceleration constraints will involve acceleration switches, which in turn implies infinite jerk. This is one of the drawbacks of TOPP with second-order constraints.

To address this issue, one can consider TOPP with third-order constraints. In many industrial applications, constraining third-order quantities such as jerk, torque rate or force rate is also part of the problem definition [124]. For instance, a hydraulic actuator exerts forces by forcing oil to travel through its piston and gets compressed, which results in the actuating force rate being restricted. As another example, DC

electric motors have bounds on input voltages, which translate directly to torque rate constraints.

5.1.1. Main Contributions

In this chapter, we follow the Numerical Integration approach and investigate the structure of the time-optimal solutions. Specifically, our contribution is threefold:

1. we propose a Multiple Shooting Method (MSM) to smoothly connect two maximum jerk profiles by a minimum jerk profile. This method is more efficient and stable than the the one proposed in [124];

2. we analyze third-order singularities, which arise very frequently and systematically stop profile integration, leading to algorithm failure. We propose a method to address such singularities;

3. based on the above contributions, we implement TOPP3, which solves TOPP with third-order constraints efficiently and yields true optimal solutions in a number of situations.

For simplicity, this chapter focuses on pure third-order constraints. Taking into account first- and second-order constraints (e.g. bounds on velocity and acceleration) is possible but will significantly increase the complexity of the exposition.

The remainder of the chapter is organized as follows. In Section 5.2, we introduce the basic notations and formulate the problem. In Section 5.3, we introduce our method to smoothly connect two maximum jerk profiles. In Section 5.4, we propose a solution to the problem of singularities. In Section 5.5, we present the numerical experiments. Finally, in Section 5.6, we offer some discussions and directions for future work.

5.2. Structure of TOPP with third-order constraints

5.2.1. Problem setting

Consider a n dof robotic system whose configuration is a vector $\mathbf{q} \in \mathbb{R}^n$. A geometric path $\mathcal{P}$ is a mapping $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ from $[0, s_{\text{end}}]$ to the configuration space. A time-parameterization of the path $\mathcal{P}$ is an increasing scalar mapping $s(t)_{t\in[0,T]}$. Differ-

entiating successively $\mathbf{q}(s(t))$ with respect to t yields

$$\begin{aligned}\dot{\mathbf{q}} =& \mathbf{q}'\dot{s} \\ \ddot{\mathbf{q}} =& \mathbf{q}''\dot{s}^2 + \mathbf{q}'\ddot{s} \\ \dddot{\mathbf{q}} =& \mathbf{q}'''\dot{s}^3 + 3\mathbf{q}''\dot{s}\ddot{s} + \mathbf{q}'\dddot{s},\end{aligned} \tag{5.1}$$

where the superscript $\dot{\square}$ denotes time-derivative and the superscript $\square'$ denotes differentiation with respect to the path position s.

From here on, we shall refer to the first, second and third time-derivatives of the path parameter s as velocity, acceleration and jerk respectively. The time-derivatives of the configuration $\mathbf{q}$ will be called joint velocity, acceleration and jerk.

The boundary conditions of a TOPP with third-order constraints consist of configuration velocities and accelerations at both the start and the goal. Combining with equation (5.1), one can compute the corresponding initial velocities and accelerations as

$$s_0 = 0, \quad \dot{s}_0 = \frac{\|\dot{\mathbf{q}}(0)\|}{\|\mathbf{q}'(0)\|}, \quad \ddot{s}_0 = \frac{\|\ddot{\mathbf{q}}(0) - \mathbf{q}''(0)\dot{s}_0^2\|}{\|\mathbf{q}'(0)\|},$$

where $\dot{\mathbf{q}}(0)$ and $\ddot{\mathbf{q}}(0)$ are the initial joint velocity and acceleration respectively. The end conditions $(s_1, \dot{s}_1, \ddot{s}_1)$ can be computed similarly.

We consider third-order constraints of the form

$$\mathbf{a}(s)\dddot{s} + \mathbf{b}(s)\dot{s}\ddot{s} + \mathbf{c}(s)\dot{s}^3 + \mathbf{d}(s) \leq 0, \tag{5.2}$$

where $\mathbf{a}(s), \mathbf{b}(s), \mathbf{c}(s), \mathbf{d}(s)$ are m-dimensional vectors.

As in the second-order case, the bounds (5.2) can represent a wide variety of constraints, from direct jerk bounds to bounds on torque rate or force rate [1]. For instance, direct jerk bounds ($\mathbf{j}_{\min} \leq \dddot{\mathbf{q}} \leq \mathbf{j}_{\max}$) can be accommodated by setting $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$ as follows

$$\mathbf{a}(s) = \begin{pmatrix} \mathbf{q}'(s) \\ -\mathbf{q}'(s) \end{pmatrix}, \qquad \mathbf{b}(s) = \begin{pmatrix} 3\mathbf{q}''(s) \\ -3\mathbf{q}''(s) \end{pmatrix},$$
$$\mathbf{c}(s) = \begin{pmatrix} \mathbf{q}'''(s) \\ -\mathbf{q}'''(s) \end{pmatrix}, \qquad \mathbf{d}(s) = \begin{pmatrix} -\mathbf{j}_{\max} \\ \mathbf{j}_{\min} \end{pmatrix}.$$

Similar to the classical TOPP algorithm with second order constraints [16, 99], one can next define, at any state $(s, \dot{s}, \ddot{s})$, the minimum jerk $\gamma(s, \dot{s}, \ddot{s})$ and maximum jerk

[1] In fact, torque rate and force rate involve an additional term $\dot{s}\mathbf{e}(s)$, but their treatment is not fundamentally different from what is presented in this chapter.

$\eta(s, \dot{s}, \ddot{s})$ as follows

$$\begin{aligned} \gamma(s, \dot{s}, \ddot{s}) &:= \max_i \left\{ \frac{-b_j(s)\dot{s}\ddot{s} - c_j(s)\dot{s}^3 - d_j(s)}{a_j(s)} \mid a_j(s) < 0 \right\}, \\ \eta(s, \dot{s}, \ddot{s}) &:= \min_j \left\{ \frac{-b_j(s)\dot{s}\ddot{s} - c_j(s)\dot{s}^3 - d_j(s)}{a_j(s)} \mid a_j(s) > 0 \right\}. \end{aligned} \tag{5.3}$$

Here the subscript j denote the j-th component of a vector. Note that the above definitions neglect the case where some a_j are zero. As in the case of second-order constraints, a path position s where at least one of the $a_i(s)$ is zero can trigger a *singularity*.

A *profile* is is a curve in the $(s, \dot{s}, \ddot{s})$-space where s is always increasing. A time-parameterization of $\mathbf{q}$ corresponds to a profile connecting $(s_0, \dot{s}_0, \ddot{s}_0)$ to $(s_1, \dot{s}_1, \ddot{s}_1)$. From the definition of γ and η, a time-parameterization of $\mathbf{q}$ satisfies the constraints (5.2) if and only if the jerk $\dddot{s}$ along the corresponding profile satisfies $\gamma \leq \dddot{s} \leq \eta$.

Next, we define a minimum jerk (respectively maximum jerk) profile as a profile along which $\dddot{s} = \gamma(s, \dot{s}, \ddot{s})$ (respectively $\dddot{s} = \eta(s, \dot{s}, \ddot{s})$). From Pontryagin's Maximum Principle, the time-optimal profile follows successively maximum and minimum jerk profiles [124]. Finding which profiles to follow and when to switch between two consecutive profiles is therefore the fundamental issue underlying TOPP with third-order constraints.

5.2.2. Introducing TOPP3

If there is no singularity, the optimal profile has a max-min-max structure [124]. Thus, to find the optimal profile, one can proceed as follows, see Fig. 5.1 for an illustration

1. integrate *forward* the maximum jerk profile (following η) from $(s_0, \dot{s}_0, \ddot{s}_0)$;
2. integrate *backward* the maximum jerk profile (following η) from $(s_1, \dot{s}_1, \ddot{s}_1)$;
3. find a minimum jerk profile that starts from one point on the first profile and ends at one point on the second one.

All profiles are integrated until failure unless otherwise specified.

To find the connecting profile of step (3), Tarkiainen and Shiller proposed to step along the first profile, integrate following minimum jerk and check whether it connects to the second profile. This exhaustive procedure is computationally inefficient and numerically unstable. We propose, in Section 5.3, a more efficient way (`bridge`) to perform the connection.

In the presence of singularities ($a_i(s) = 0$), the max-min-max procedure just described will fail, irrespective of the connection method. This is because the integrated profiles diverge as they approach a singularity and quickly terminate, see Fig. 5.3A.

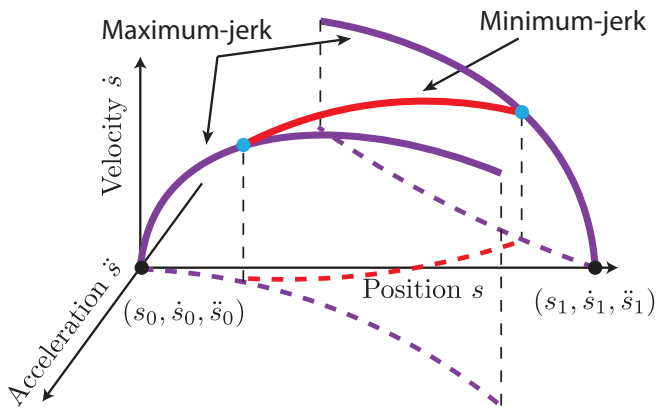


Figure 5.1.: An optimal parametrization with max-min-max structure.

This behavior is the same as in the second-order case [99]. To address this issue, we propose, in Section 5.4, a method (`extend`) that allows extending profiles *through* singularities. Note that the structure of the final profile will no longer be max-min-max but max-min-max-...-min-max.

The above discussion can be encapsulated into Algorithm 2, which we call *TOPP3*. In the next sections, we shall discuss in detail the two main components of TOPP3, namely `bridge` and `extend`.

Algorithm 2: TOPP3

Input : Initial and final conditions $(s_0, \dot{s}_0, \ddot{s}_0)$, $(s_1, \dot{s}_1, \ddot{s}_1)$
Output: The optimal profile P_{opt}

$P_{\text{forward}} \leftarrow$ integrate forward from $(s_0, \dot{s}_0, \ddot{s}_0)$ following maximum jerk until termination
while *found next singularity* **do**
  $P_{\text{forward}} \leftarrow$ `extend`$(P_{\text{forward}}$, *singularity*$)$
end
$P_{\text{backward}} \leftarrow$ integrate backward from $(s_1, \dot{s}_1, \ddot{s}_1)$ following maximum jerk until termination
while *found next singularity* **do**
  $P_{\text{backward}} \leftarrow$ `extend`$(P_{\text{backward}}$, *singularity*$)$
end
$P_{\text{opt}} \leftarrow$ `bridge`$(P_{\text{forward}}, P_{\text{backward}})$

5.2.3. Remarks

Optimality of the connection

In the absence of singularities, the max-min-max structure described earlier is only a necessary condition for optimality and not a sufficient one. Theoretically, to find

the true optimal solution, it suffices to enumerate all candidates having max-min-max structure and select the one with the shortest time duration.

Next, since the first and last maximum jerk profiles are well-determined (their starting (respectively ending) conditions are given), different solutions only differ by the connecting minimum jerk profile. If there exists one unique minimum jerk profile that can connect the two maximum jerk profiles, then the corresponding solution is automatically the optimal one. In practice, we have never encountered the case when there are more than one possible connecting minimum jerk profile. Note that Tarkiainen and Shiller [124] implicitly assumed that there exists one unique connecting minimum jerk profile.

Other switching structures

The TOPP problem with second-order constraints involves three types of switch points, i.e., points where the optimal profile changes from minimum acceleration to maximum acceleration and vice-versa (cf. [99]): (a) singular (b) discontinuous and (c) tangent points. These switch points, in general, lie on the Maximum Velocity Curve (MVC).

We argue that TOPP with third-order constraints similarly involves singular, discontinuous and tangent switch points, and that those points, in general, lie on the Maximum and Minimum Acceleration Surfaces (MaAS and MiAS).

In Section 5.4, we shall discuss the singular switch points, which are the most commonly encountered and harmful (if inappropriately treated) switch points. As in the case of second-order constraints, discontinuous switch points can always be avoided if the path is sufficiently smooth. Finally, tangent switch points are left for future work. Let us simply note here that they occur much less frequently than singular switch points.

Another type of switching structure arises when constraints of different orders interact. For instance, in the classic TOPP problem, when an integrated profile hits a first-order constraint (e.g. direct velocity bound), then it must "slide" along the boundary defined by that first-order constraint, giving rise to a different type of switch point. We envisage that such switching behavior can also happen when first, second, and third-order constraints interact. The study of these interactions is also left for future work.

5.3. Connecting profiles using Multiple Shooting

Instead of the exhaustive search suggested in [124], we propose here a method, termed `bridge`, which is based on Multiple Shooting to find a minimum jerk profile that can connect two maximum jerk profiles. Specifically, consider the problem of connecting

two maximum jerk profiles $\mathcal{M}$ and $\mathcal{N}$. We define a potential solution Γ as a $(2N+4)$-dimensional vector

$$\Gamma := [\dot{s}_0, \ddot{s}_0, ..., \dot{s}_N, \ddot{s}_N, s_\mathcal{M}, s_\mathcal{N}],$$

where $\dot{s}_i, \ddot{s}_i$, $i \in [0, N]$ are the guessed velocities and accelerations at the i-th point and $(s_\mathcal{M}, s_\mathcal{N})$ are the guessed starting and ending positions on profiles $\mathcal{M}$ and $\mathcal{N}$ respectively. We consider a uniform grid, i.e.

$$s_i := s_\mathcal{M} + \frac{i}{N}(s_\mathcal{N} - s_\mathcal{M}).$$

Next, assume that we integrate following minimum jerk from $(s_i, \dot{s}_i, \ddot{s}_i)$ until s_j. We define the function $X : \mathbb{R}^4 \to \mathbb{R}^2$ as

$$X(s_i, \dot{s}_i, \ddot{s}_i, s_j) := (\dot{s}_j, \ddot{s}_j),$$

where $\dot{s}_j, \ddot{s}_j$ are the corresponding velocity and acceleration at s_j. This allows us to define the defect function by

$$F(\Gamma) := \begin{pmatrix} X(s_0, \dot{s}_0, \ddot{s}_0, s_1) - [\dot{s}_1, \ddot{s}_1]^T \\ X(s_1, \dot{s}_1, \ddot{s}_1, s_2) - [\dot{s}_2, \ddot{s}_2]^T \\ \cdots \\ X(s_{N-1}, \dot{s}_{N-1}, \ddot{s}_{N-1}, s_N) - [\dot{s}_N, \ddot{s}_N]^T \\ r_\mathcal{M}(s_\mathcal{M}) - [\dot{s}_0, \ddot{s}_0]^T \\ r_\mathcal{N}(s_\mathcal{N}) - [\dot{s}_N, \ddot{s}_N]^T \end{pmatrix}, \tag{5.4}$$

where $r_\mathcal{M}(s_\mathcal{M})$ (respectively $r_\mathcal{N}(s_\mathcal{N})$) are the velocity and acceleration on profile $\mathcal{M}$ (respectively $\mathcal{N}$) at position $s_\mathcal{M}$ (respectively $s_\mathcal{N}$).

The connection problem can now be formulated as

$$\begin{aligned} \text{solve} \quad & F(\Gamma) = 0 \\ \text{subject to} \quad & s_{\mathcal{M},beg} \leq s_\mathcal{M} \leq s_{\mathcal{M},end}, \\ & s_{\mathcal{N},beg} \leq s_\mathcal{N} \leq s_{\mathcal{N},end}, \end{aligned} \tag{5.5}$$

where $(s_{\mathcal{M},beg}, s_{\mathcal{M},end})$ denote positions of profile $\mathcal{M}$'s end points (respectively profile $\mathcal{N}$).

To solve (5.5), we employ the Newton method. Although the problem is non-linear and non-convex, the algorithm still converges very quickly to a solution. Fig. 5.2 shows a particular instance where a solution can be found in 4 iterations.

Regarding computational cost, we found that it correlates to the magnitude of jerk bounds. For instance, at $1000\,\text{rads}^{-3}$ a `bridge` function call takes only $10\,\text{ms}$ while at

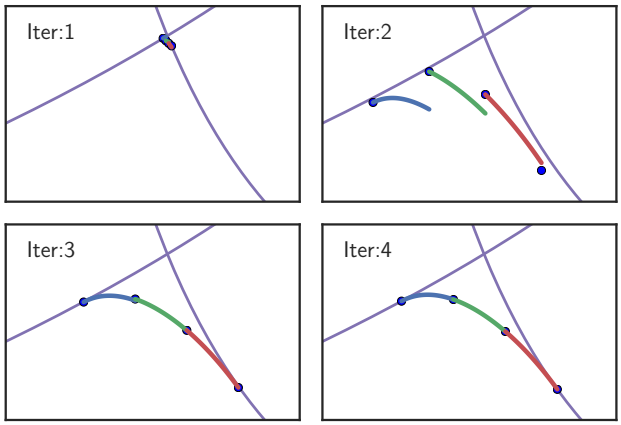


Figure 5.2.: Using MSM to find a minimum jerk profile connecting two maximum jerk profiles (purple). Note that the profiles are in 3D but projected to 2D for convenience. The number of segments is $N = 3$. The algorithm took 4 iterations to converge.

$100\,\mathrm{rads}^{-3}$, it is approximately $200\,\mathrm{ms}$. How jerk bounds precisely affect computation time is left for future investigations.

5.4. Characterizing and addressing singularities

Singularities in the third-order case are very similar to those in the second-order case: (i) they arise at positions s where the minimum and maximum jerk γ and η cannot be properly defined because of the division by $a_k(s) = 0$ for some k; (ii) they cause profiles to terminate prematurely, causing algorithm failure (see Fig. 5.3). In the sequel, we discuss how to characterize third-order singularities and how to address singularities, taking much inspiration from the second-order case [99].

5.4.1. Maximum and Minimum Acceleration Surfaces

In the second-order case, singularities mostly appear on the Maximum Velocity Curve (MVC) [99, 94, 112]. Here, we define the Maximum and Minimum Acceleration Surfaces (MaAS and MiAS), which are the third-order counterparts of the MVC.

Definition 10 (MaAS/MiAS)**.** Consider the set of feasible accelerations

$$\mathcal{J}(s, \dot{s}) := \{\ddot{s} \mid \exists \dddot{s}, \mathbf{a}(s)\dddot{s} + \mathbf{b}(s)\dot{s}\ddot{s} + \mathbf{c}(s)\dot{s}^3 + \mathbf{d}(s) \leq 0\}.$$

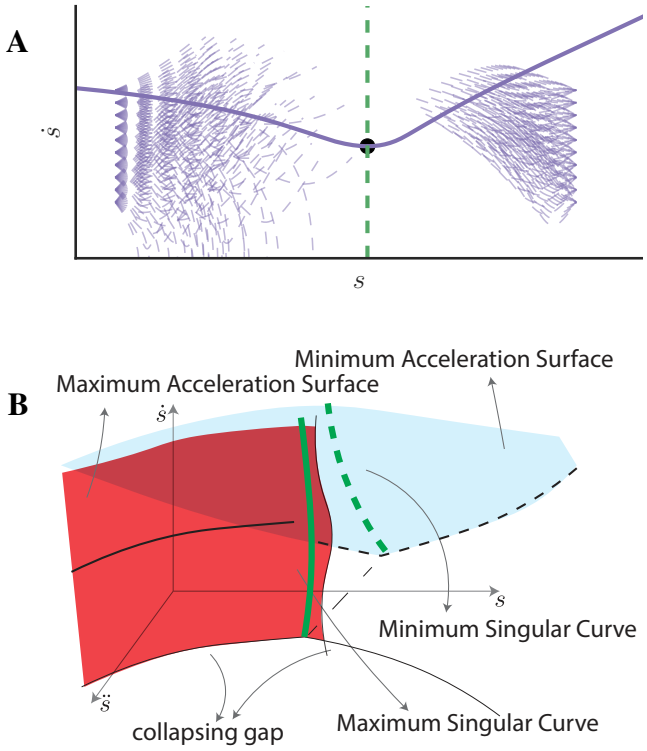


Figure 5.3.: **A**: Profiles approaching singularities (purple dashed lines) diverge and terminate early. The diverging directions include both vertical (along $\dot{s}$ axis) and lateral (along $\ddot{s}$ axis, which is not shown). However, profiles (purple solid lines) starting from a point lying between the singular curves (green dashed lines) are not affected by the singularities. **B**: Singularities consist of maximum or minimum singular curves lying on the maximum or minimum acceleration surfaces (MaAS/MiAS) respectively.

We define the MaAS and MiAS by

$$\begin{aligned} \mathrm{MaAS}(s, \dot{s}) &:= \max_{\ddot{s} \in \mathcal{J}(s,\dot{s})} \ddot{s} \\ \mathrm{MaAS}(s, \dot{s}) &:= \min_{\ddot{s} \in \mathcal{J}(s,\dot{s})} \ddot{s}. \end{aligned} \tag{5.6}$$

If $\mathcal{J}(s, \dot{s})$ is empty then the surfaces are not defined at $(s, \dot{s})$. ◊

This definition does not use the maximal controls γ and η but directly uses the constraints (5.2). The advantage is that even in the presence of a constraint k such that $a_k(s) = 0$, the surfaces are still well-defined. Additionally, on both surfaces, the maximum and minimum jerks are equal almost everywhere except on singular curves (see Prop. 4 in the Appendix).

5.4.2. Characterizing third-order singularities

Definition 11 (Singular curve)**.** We say that the k-th constraint triggers a singularity at s^* if $a_k(s^*) = 0$ and the set $C(s^*)$ defined by

$$\begin{aligned} C(s^*) := \{(s^*, \dot{s}, \ddot{s}) \mid \exists \dddot{s} : \\ a_k(s^*)\dddot{s} + b_k(s^*)\dot{s}\ddot{s} + c_k(s^*)\dot{s}^3 + d_k(s^*) = 0, \\ a_j(s^*)\dddot{s} + b_j(s^*)\dot{s}\ddot{s} + c_j(s^*)\dot{s}^3 + d_j(s^*) \leq 0, j \neq k \end{aligned} \Big\} \tag{5.7}$$

is non-empty. We say $C(s^*)$ is the singular curve at s^*. ◊

We show in the Appendix that all singular curves lie on either the MaAS or MiAS (Prop. 6). Furthermore, a singular curve lies on the MaAS if $b_k(s^*) > 0$ and on the MiAS if $b_k(s^*) < 0$. We shall also refer to singular curves on MaAS and MiAS as maximum and minimum singular curves respectively.

As in the second-order case, we can note the following behaviors:

- Forward integrations following maximum jerk and backward integrations following minimum jerk diverge when they approach a minimum singular curve;
- Forward integrations following minimum jerk and backward integrations following maximum jerk diverge when they approach a maximum singular curve;
- Forward and backward integrations following either maximum jerk or minimum jerk *starting from* a feasible point at s^* are not affected by the singular curves.

To understand these observations, we note that at fixed velocity $\dot{s} = \dot{s}_0$, constraints (5.7) has the same form as second order-constraints

$$\mathbf{a}_{\text{2nd}}(s)\ddot{s} + \mathbf{b}_{\text{2nd}}(s)\dot{s}^2 + \mathbf{c}_{\text{2nd}}(s) \leq 0, \tag{5.8}$$

where $\mathbf{a}_{\text{2nd}}, \mathbf{b}_{\text{2nd}}, \mathbf{c}_{\text{2nd}}$ are the corresponding vectors. In the second-order case, constraints (5.8) cause integrations to diverge. It therefore suggests that integrations (third-order) projected on to a fixed velocity surface would likely diverge, which is consistent with our observations. A more rigorous analysis is left for future work.

To compute a singular curve C at s^*, we simply find the maximum and minimum velocities $\dot{s}^*_{\text{max}}, \dot{s}^*_{\text{min}}$ of each curve and compute $\ddot{s}^*$ by the equality constraint in Eq. (5.7). This procedure correctly returns C because a singular curve is connected (See Prop. 3 in the Appendix).

To compute $\dot{s}^*_{\max}, \dot{s}^*_{\min}$ one needs to solve a pair of linear programming problems

$$\begin{aligned} &\text{maximize} && [0,0,1]^T[\dddot{s}, \dot{s}\ddot{s}, \dot{s}^3] \\ &\text{subject to} && (5.7) \end{aligned}$$

and

$$\begin{aligned} &\text{maximize} && [0,0,-1]^T[\dddot{s}, \dot{s}\ddot{s}, \dot{s}^3] \\ &\text{subject to} && (5.7). \end{aligned}$$

5.4.3. Extending a profile through a singularity

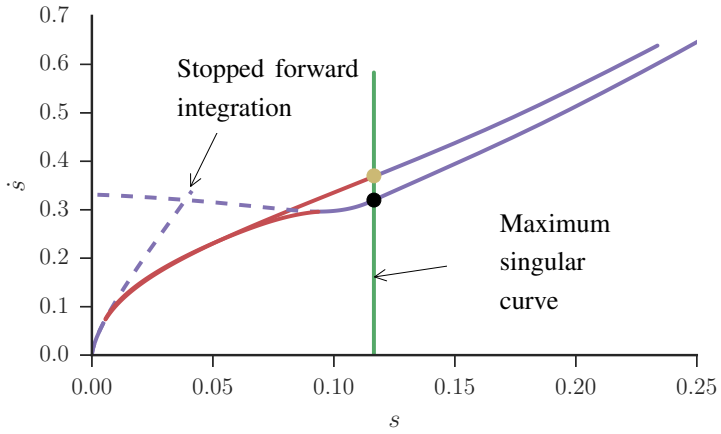


Figure 5.4.: Comparison of the conjectured time-optimal extension (via the yellow point) with an extension computed by picking a feasible point lying at s^* (black point) and uses it to extend the forward integration. Note that the profiles are projected onto the $(s, \dot{s})$ plane.

We now describe a method, termed `extend`, to extend a forward maximum jerk profile through a singularity s^*. As noted before, integrating forward following maximum jerk *from* s^* is not problematic, so the main difficulty consists of connecting backwards to the initial forward profile.

Note first that there exists a *family* of possible backward connections, obtained as below

1. choose a feasible point at s^* (i.e. a point $(s^*, \dot{s}, \ddot{s})$ for which there exists a feasible jerk);
2. integrate backward following maximum jerk from this point;
3. connect the original forward maximum jerk profile to the new backward maximum jerk profile by the `bridge` procedure of Section 5.3.

We conjecture that, within this family of possible connections, the *optimal* connection has the following properties

1. the starting point at s^* belongs to the *maximum singular curve*;

2. there is no backward maximum jerk profile: in other words, the connecting minimum jerk profile starts directly at s^*;

3. as a consequence, the min-to-max switch happens directly at s^* and not before, as in the generic case above.

See Fig. 5.4 for an illustration.

Accordingly, we propose the following strategy to perform the backward connection using MSM. Using subscript A and C to denote the forward profile and the singular curve, we define a solution $\Gamma \in \mathbb{R}^{2N+4}$ as

$$\Gamma := [\dot{s}_0, \ddot{s}_0, ..., \dot{s}_N, \ddot{s}_N, s_\mathrm{A}, \dot{s}_\mathrm{C}],$$

where $\dot{s}_i$ and $\ddot{s}_i$ are the velocity and acceleration at s_i, s_A and $\dot{s}_\mathrm{C}$ are the guessed starting position and ending velocity on A and C respectively. This gives the defect function

$$F(\Gamma) := \begin{pmatrix} X(s_1, \dot{s}_1, \ddot{s}_1, s_0) - [\dot{s}_0, \ddot{s}_0]^T \\ X(s_2, \dot{s}_2, \ddot{s}_2, s_1) - [\dot{s}_1, \ddot{s}_1]^T \\ \cdots \\ X(s_N, \dot{s}_N, \ddot{s}_N, s_{N-1}) - [\dot{s}_{N-1}, \ddot{s}_{N-1}]^T \\ r_\mathrm{A}(s_A) - [\dot{s}_0, \ddot{s}_0]^T \\ \bar{r}_\mathrm{C}(\dot{s}_\mathrm{C}) - [\dot{s}_N, \ddot{s}_N]^T \end{pmatrix}, \tag{5.9}$$

where $r_\mathrm{A}(s_\mathrm{A})$ and $\bar{r}_\mathrm{C}(s_\mathrm{C})$ give the velocity and acceleration on the profile and the curve. Again, Newton's method can be employed to find the root of (5.9).

There exists an additional difficulty with respect to Section III: the optimal jerk is ill-defined on the singular curves because of a division by zero: $a_k(s^*) = 0$. Taking again inspiration from [99], we can show that the optimal jerk on the singular curves is in fact given by the following *singular jerk*

$$\dddot{s}_\mathrm{sglr} = -\frac{d_k'(s^*) + c_k'(s^*)\dot{s}^3 + 3c_k(s^*)\dot{s}\ddot{s} + b(s^*)(\dot{s}\ddot{s} + \ddot{s}^2/\dot{s})}{a_k'(s^*) + b_k(s^*)}. \tag{5.10}$$

This expression can be derived similarly as in [99].

Fig. (5.4) compares the conjectured optimal connection with a generic connection. One can note that the conjectured profile has a higher velocity at *any position*, which implies time-optimality.

The case of a backward extension can be treated similarly.

Table 5.1.: Kinematic limits for Denso robot arm

Limits	J1	J2	J3	J4	J5	J6
Vel (rads^{-1})	3.92	2.61	2.85	3.92	3.02	6.58
Accel (rads^{-2})	19.7	16.8	20.7	20.9	23.7	33.5
Jerk (rads^{-3})	100.	100.	100.	100.	100.	100.

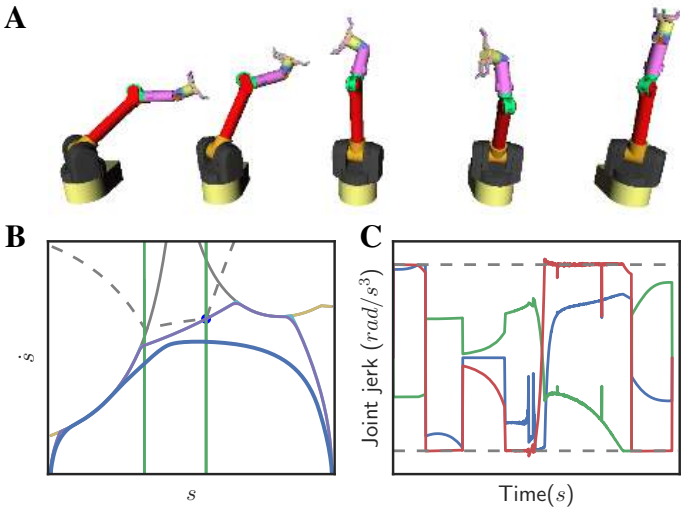


Figure 5.5.: **A**: Snapshots of the robot. **B**: The profile constrained at $100\,\mathrm{rad/s^3}$ (blue), the profile constrained at $1000\,\mathrm{rad/s^3}$ (purple) and the profile with unconstrained jerk (light blue). Singularities are plotted as green lines. Note that the profiles are projected onto the $(s, \dot{s})$ plane. **C**: Joint jerks versus time.

5.5. Simulation results

5.5.1. Simulation results

We implemented and tested TOPP3 on a random geometric path subjecting to constraints on joint velocities, accelerations and jerks (Table 5.1). Implementation of acceleration and velocity constraints follows [99].

Different scenarios were considered. In the first one, we set restrictive bounds on joint jerks at $100\,\mathrm{rad/s^3}$. There were several singularities which are plotted as green lines in Fig. 5.5. Despite the existence of these singularities, TOPP3 was able to compute the time-optimal parametrization. We note that TOPP3 extended the forward profile once via the maximum singular curve on the left.

Next, we considered a more practical set of bounds at $1000\,\mathrm{rad/s^3}$ on joint jerks. In this case, TOPP3 also terminated successfully. Moreover, we found that singularities did not affect the profiles: the final profile was computed without having to extend any profile via any singularities.

Lastly, we compared these profiles with one that does not subject to bounds on

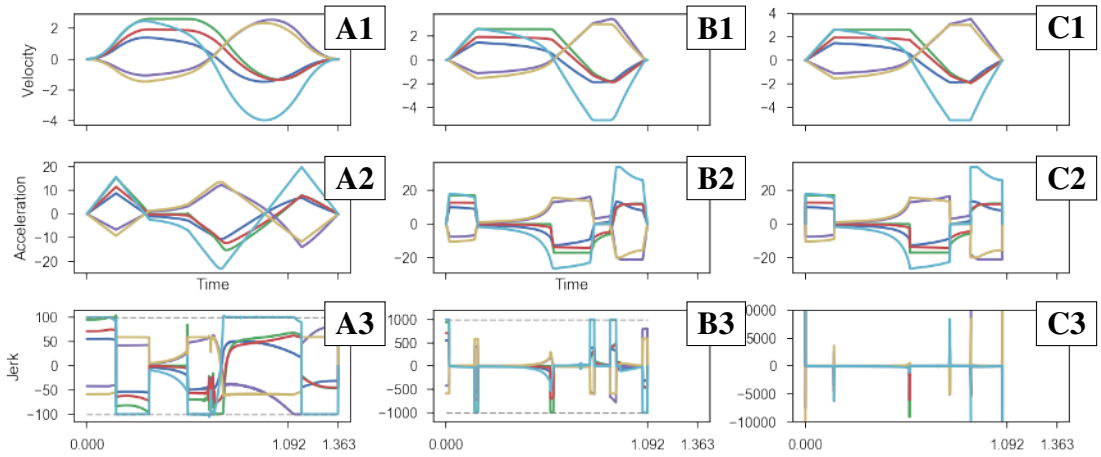


Figure 5.6.: Comparing the resulting trajectories from TOPP3 with $100\,\mathrm{rad/s^3}$ bound on joint jerks (**A1-3**), $1000\,\mathrm{rad/s^3}$ bound on joint jerks (**B1-3**) and one from TOPP without any bound on jerk (**C1-3**). Note that, when there is no bound on jerk, the latter reaches extremely high values (C3).

joint jerks (computed with the original TOPP algorithm). We observed that the profile bounded at $100\,\mathrm{rad/s^3}$ appears to be a smoothed version of the profile bounded at $1000\,\mathrm{rad/s^3}$, which in turn is a smoothed version of the one not subject to any jerk bound (Fig. 5.5).

Fig. 5.6 shows the joint trajectories computed from these profiles. We note that the profiles are bang-bang, satisfy all kinematic constraints and have different total duration. In particular, they last $1.36\,\mathrm{sec}$, $1.19\,\mathrm{sec}$ and $1.18\,\mathrm{sec}$. Note that, when there is no bound on jerk, the latter reaches extremely high values (Fig. 5.6-C3).

Table 5.2 reports computation time for the three scenarios respectively. The experiments were ran on a single core at $2.00\,\mathrm{GHz}$ and $8\,\mathrm{Gb}$ of memory. We remark that the pre-processing – computation of singularities and switch points – is written in Python and is significantly slower than the integration and MSM procedures which are written in Cython. Therefore, we only compared the total time* which neglects pre-processing. In this regard, computing the first and second profiles took 83 times and 19 times longer than the third one.

5.5.2. Singularities caused by third-order constraints

Third-order singularities appear quite frequently, almost as frequent as singular switch points which are many [99]. In the Fig. 5.5, one can see that the singular curves (green lines) correspond to points where the MVC is continuous but is not differentiable, which are singular switch points [99].

We observed that in the second scenario ($1000\,\mathrm{rad/s^3}$), singularities did not affect integrations. We found that this phenomenon only appears when the bounds on joint

Table 5.2.: Computation time for different levels of bound on joint jerks

Jerk bound $(\mathrm{rad/s^{-3}})$	100	1000	∞
Time step (ms)	1	1	1
Total (ms)	754	181	72
`integrate`	4.05	2.36	2.24
`bridge`	364	42.2	0.3
`extend`	134	73.2	3.50
Pre-process	243	63.7	65.9
Total*[2] (ms)	511	118	6.1
	83x	19x	1x

jerks are high in comparison with the bounds on joint accelerations. Here is one possible explanation: at high jerk bounds, the singular curves have high acceleration. However, as second-order constraints restrict integrations from having high acceleration, the integrations will not come close to these curves. Now, as analyzed, we observe that integrations only diverge near to the singular curves; therefore, these integrations are not affected. The precise conditions at which this happens is left for future work.

5.6. Summary

In this chapter, we have studied the structure of the Time-Optimal Path Parameterization (TOPP) problem with third-order constraints. We have argued that the optimal profile can be obtained by integrating alternatively maximum and minimum jerk, yielding a max-min-max-...-min-max structure. We have identified two main difficulties in the integration process: (i) how to smoothly connect two maximum jerk profiles by a minimum jerk profile, and (ii) how to extend the integration through singularities which, if not properly addressed, would systematically cause integration failure. We proposed some solutions to these two difficulties and, based on these solutions, implemented an algorithm – TOPP3 – to solve the TOPP problem with third-order constraints in a number of representative scenarios.

There are still a number of open theoretical and practical questions, which we are currently actively investigating. For instance,

1. Under which condition the profile returned by the connection procedure (`bridge`) is unique? Alternatively, can we enumerate all connecting profiles?

2. Similarly, under which condition the profile returned by the extension procedure (`extend`) is unique?

[2]Not including time to compute singularities and switch points.

3. How to characterize and address the tangent switch points (which probably exist in the form of tangent *curves*)?

Addressing all these (difficult) questions will enable us to implement TOPP3 in a fast and robust manner, which in turn can be useful for a wide range of robotics applications.

5.7. Additional proofs and remarks

5.7.1. Some proofs regarding third-order singularities

Proposition 1. *All singular curve lies on either the MaAS or the MiAS. Furthermore, a singular curve lies on the MaAS if* $b_k(s^*) > 0$ *and on the MiAS if* $b_k(s^*) < 0$.

We assume $b_k(s^*) \neq 0$ since it is fairly rare for both a_k and b_k to become zero.

Proof. We will prove the first proposition by contradiction. Consider a singular curve C triggered by the k-th constraint, there exists a point $(s^*, \dot{s}^*, \ddot{s}^*) \in C$ that do not lie on either the MaAS or MiAS. That is

$$\mathrm{MaAS}(s^*, \dot{s}^*) < \ddot{s}^* < \mathrm{MaAS}(s^*, \dot{s}^*).$$

It follows that we can always find $\ddot{s}_1$ and $\ddot{s}_2$ such that

$$\mathrm{MaAS}(s^*, \dot{s}^*) < \ddot{s}_2 < \ddot{s}^* < \ddot{s}_1 < \mathrm{MaAS}(s^*, \dot{s}^*).$$

Now, by equation (5.7), we have the equality

$$b_k(s^*)\dot{s}^*\ddot{s}^* + c_k(s^*)\dot{s}^{*3} + d_k(s^*) = 0, \tag{5.11}$$

since $a_k(s^*) = 0$. Now, if $b_k(s^*) > 0$, we replace $\ddot{s}^*$ with $\ddot{s}_1$ and notice that $\dot{s} > 0$ to see that equation (5.11) is strictly positive. Similarly if $b_k(s^*) < 0$ then we replace $\ddot{s}^*$ with $\ddot{s}_2$ to see that equation (5.11) is strictly negative. Both are contradictions. We neglect the case where $b_k(s^*) = 0$.

Finally, to prove the second proposition, we simply remark that if $b_k(s^*) > 0$, then there must not exists any feasible $\ddot{s}_1$ that is greater than $\ddot{s}^*$. It therefore follows that

$$\ddot{s}^* = \mathrm{MaAS}(s^*, \dot{s}^*).$$

The case where $b_k(s^*) < 0$ is proven similarly. □

Proposition 2. *The maximum and minimum jerks are equal almost everywhere on the MaAS and the MiAS except on singular curves.*

Proof. Considering a point at $(s_0, \dot{s}_0, \ddot{s}_0)$ on the MaAS which does not lie on any singular curve, by definition, $\ddot{s}_0$ is the maximum acceleration for $s = s_0, \dot{s} = \dot{s}_0$ subjecting to constraints (5.7). Since all constraints (5.7) are linear the set of feasible $(\ddot{s}, \dddot{s})$ is a convex polygon on the plane.

Now, since this polygon does not contain any edge that is parallel to the $\ddot{s}$-axis ($(s_0, \dot{s}_0, \ddot{s}_0)$ does not lie on any singular curve), $\ddot{s} = \ddot{s}_0$ is maximized at a single vertex of the polygon. Therefore there is only one feasible jerk for $\ddot{s} = \ddot{s}_0$.

A similar proof can be written for the MiAS. □

Proposition 3. *Singular curves are connected.*

Proof. Consider a singular curve C in the $(s, \dot{s}, \ddot{s})$ space triggered by the k-th constraint. Let the convex set defined by the feasibility condition (5.7) be $\mathcal{A}$ and the mapping from $\mathcal{A}$ using the equality

$$b_k(s^*)\dot{s}\ddot{s} + c_k(s^*)\dot{s}^3 + d_k(s^*) = 0.$$

to the singular curve C be f. By definition, $f(\mathcal{A}) = C$.

Now, since $\mathcal{A}$ is convex and thus connected and f is also continuous for $b_k(s^*) \neq 0, \dot{s}^* \neq 0$, using Theorem 4.22 in [106], it follows that C is connected. □

Chapter 6.

Time-Optimal Path Tracking via Reachability Analysis

6.1. Introduction

Time-optimal motion planning and control along a predefined path are fundamental and important problems in robotics, motivated by many industrial applications, ranging from machining, to cutting, to welding, to painting, etc.

The *planning* problem is to find the Time-Optimal Path Parameterization (TOPP) of a path under kinematic and dynamic bounds. The underlying assumptions are that the robot is perfectly modeled, no perturbations during execution and no initial tracking errors. This problem has been extensively studied since the 1980's [16], see [99, 100] for recent reviews.

The *control* problem, which looks for a control strategy to time-optimally track the path while *accounting for model inaccuracies, perturbations and initial tracking errors*, is comparatively less well understood. We refer to this problem as the Time-Optimal Path Tracking problem, or "path tracking problem" in short.

6.1.1. Contributions of the chapter

To guarantee that the path controller will always find feasible controls requires a certain level of foresight: one needs to take into account *all possible perturbations* along *the path*. In this chapter, we build on the recent formulation of TOPP by Reachability Analysis [96] to provide such foresight. Specifically, we compute sets of robust con-

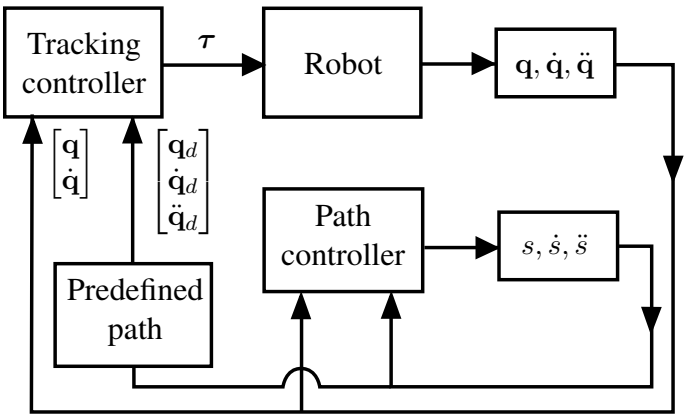


Figure 6.1.: Block diagram of an Online Scaling controller.

trollable states [1] that guarantee the existence of feasible controls for bounded tracking errors. From these sets, a class of path tracking controllers that have exponential stability and feasiblity guarantees is identified. The time-optimal controller is then found straightforwardly.

The rest of the chapter is organized as follows. Section 6.2 provides the background on the path tracking problem and the path tracking controller. Section 6.3 presents the main contributions. Section 6.4 reports experimental results, demonstrating the effectiveness of the proposed approach. Finally, Section 6.5 delivers concluding remarks and sketches directions for future research.

6.1.2. Notation

We adopt the following conventions. Vectors are denoted by bold letters: $\mathbf{x}$. The i-th component of a vector is denoted using subscript i: x_i. A vector quantity at stage j is denoted by bold letters with subscript j: $\mathbf{x}_j$, its i-th component is denoted by adding a second subscript i: x_{ji}. Define function $\phi(s, s^d; \mathbf{p}) := [\mathbf{p}(s)^\top, \mathbf{p}'(s)^\top s^d]^\top$, argument $\mathbf{p}$ will be neglected if clear from context. If $\mathbf{x}, \mathbf{y}$ are two vectors, $(\mathbf{x}, \mathbf{y})$ denote the concatenated vector $(\mathbf{x}^\top, \mathbf{y}^\top)^\top$. Values of differential quantities, such as $\dot{\mathbf{q}}$, have superscript d: $\mathbf{q}_0^d$.

[1] These are parameterization states, which are defined as squared path velocities. In Section 6.3, precise definitions are given.

6.2. Background: Path Tracking problem and controllers

6.2.1. Path Tracking problem

Path tracking is the problem of designing a controller to make the robot's joint positions follow a path parameterization of a predefined path. The path parameterization is not fixed but is generated by the controller in an online manner.

Specifically, we consider a n-dof manipulator with the dynamic equation

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{h}(\mathbf{q}) = \tau, \tag{6.1}$$

where $\mathbf{q} \in \mathbb{R}^n$ and $\tau \in \mathbb{R}^n$ denote the vectors of joint positions and joint torques; $\mathbf{M}, \mathbf{C}, \mathbf{h}$ are appropriate functions. The joint torques are bounded:

$$\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{\max}. \tag{6.2}$$

A geometric path is a twice-differentiable function $\mathbf{p}(s)_{s\in[0,1]} \in \mathbb{R}^n$. Notice that in contrast with the previous chapter, here a different letter is used to denote the path, distinguished with from the trajectory. We make this choice to improve the clarify of later presentation. A *path parameterization* is a twice-differentiable non-decreasing function $s(t)_{t\in[0,T]} \in [0, 1]$. Path parameterizations are also subject to terminal velocity constraints of the form $\dot{s}(T) \in \mathbb{I}_{\text{end}}$, where $\mathbb{I}_{\text{end}}$ is called the terminal set. To generate the parameterization, one directly controls the path acceleration. Let u denote the path parameterization control: $\ddot{s} = u$.

The state-space equation of the *coupled system* consisting of the manipulator and the path parameterization reads

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\\ s\\ \dot{s}\end{bmatrix} = \begin{bmatrix}\dot{\mathbf{q}}\\ \mathbf{M}(\mathbf{q})^{-1}(\boldsymbol{\tau} - \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} - \mathbf{h}(\mathbf{q}))\\ \dot{s}\\ u\end{bmatrix}. \tag{6.3}$$

Let $\mathbf{y}$ denote the state of the coupled system $[\mathbf{q}, \dot{\mathbf{q}}, s, \dot{s}]$, Eq. (6.3) can be written concisely as

$$\dot{\mathbf{y}} = f(\mathbf{y}) + g(\mathbf{y})\begin{bmatrix}\boldsymbol{\tau}\\ u\end{bmatrix}.$$

Consider a control law $[\boldsymbol{\tau}, u] = \pi(\mathbf{y})$ that always satisfies the torque bounds, one

obtains the autonomous dynamics

$$\dot{\mathbf{y}} = f(\mathbf{y}) + g(\mathbf{y})\pi(\mathbf{y}) = \hat{f}(\mathbf{y}). \tag{6.4}$$

We say that $\mathbf{y}(t)_{t\in[0,T]} \in \mathbb{R}^{2n+2}$ is a *solution* of Eq. (6.4) if

$$\dot{\mathbf{y}}(t) = \hat{f}(\mathbf{y}(t)),\ y_{2n+1}(t) \in [0,1],\ \forall t \in [0,T].$$

It is a *feasible solution* if additionally,

$$\begin{aligned} &y_{2n+2}(t) \geq 0,\ \forall t \in [0,T], \\ &y_{2n+1}(T) = 1, y_{2n+2}(T) \in \mathbb{I}_{\text{end}}. \end{aligned}$$

It is a solution *with initial value* $\mathbf{y}_0$ if $\mathbf{y}(0) = \mathbf{y}_0$.

The coupled system is *stable* at $(s_0, s_0^d) \in [0,1] \times [0,\infty]$ if for any $R > 0$, there exist $r > 0$ such that if $\|(\mathbf{q}_0, \mathbf{q}_0^d) - \phi(s_0, s_0^d; \mathbf{p})\|_2 \leq r$, the solution $\mathbf{y} =: (\mathbf{q}, \dot{\mathbf{q}}, s, \dot{s})$ with initial value $(\mathbf{q}_0, \mathbf{q}_0^d, s_0, s_0^d)$ exists and

$$\|(\mathbf{q}(t), \dot{\mathbf{q}}(t)) - \phi(s(t), \dot{s}(t))\|_2 < R,\ \forall t \in [0,T].$$

See Section 6.1.2 for the definition of the function ϕ. The coupled system is *exponentially stable* at $(s_0, s_0^d) \in [0,1] \times [0,\infty]$ if it is stable and there exist $r_e > 0$ such that if $\|(\mathbf{q}_0, \mathbf{q}_0^d) - \phi(s_0, s_0^d; \mathbf{p})\|_2 \leq r_e$, the solution $\mathbf{y} = (\mathbf{q}, \dot{\mathbf{q}}, s, \dot{s})$ with initial value $(\mathbf{q}_0, \mathbf{q}_0^d, s_0, s_0^d)$ exists and satisfies

$$\|(\mathbf{q}(t), \dot{\mathbf{q}}(t)) - \phi(s(t), \dot{s}(t))\|_2 < Ke^{-\lambda t}, 0 \leq t \leq T,$$

for some positive real numbers K, λ.

The idea of agumenting to the path tracking problem the path velocity and acceleration variableshas been proposed earlier in the literature, in the context of non time-optimal path tracking [14, 39] or nonlinear trajectory tracking [80].

6.2.2. Path Tracking controllers

A path tracking controller consists of a path controller, which controls the path acceleration to generate a desired joint trajectory $\mathbf{q}_d(t) := \mathbf{p}(s(t))$, and a tracking controller that controls the joint torques to track the desired joint trajectory.

A common control objective is to track a predefined reference path parameterization. Here we consider the time-optimal objective, which is to traverse the path as fast as possible.

Similar to [33] and subsequent developments presented in [32, 14], we employ the computed-torque trajectory tracking scheme for the tracking controller. This scheme implements the following control law

$$\begin{aligned}\boldsymbol{\tau} =& \mathbf{M}(\mathbf{q})[\ddot{\mathbf{q}}_d + \mathbf{K}_p\mathbf{e} + \mathbf{K}_d\dot{\mathbf{e}}] \\ &+ \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{h}(\mathbf{q}),\end{aligned} \tag{6.5}$$

where $\mathbf{e}$ denote the joint positions error vector, defined as $\mathbf{e} := \mathbf{q}_d(t) - \mathbf{q}(t)$, $\mathbf{K}_p$ and $\mathbf{K}_d$ are the PD gain matrices. The vector $(\mathbf{e}, \dot{\mathbf{e}})$ is called the tracking error. The first and second time derivatives of the desired joint trajectory, which are used in (6.5), are given by

$$\begin{aligned}\dot{\mathbf{q}}_d(t) &= \mathbf{p}'(s(t))\dot{s}(t), \\ \ddot{\mathbf{q}}_d(t) &= \mathbf{p}'(s(t))\ddot{s}(t) + \mathbf{p}''(s(t))\dot{s}(t)^2.\end{aligned}$$

Rearranging Eq. (6.5), one obtains a formula for joint torques:

$$\boldsymbol{\tau} = \hat{\mathbf{a}}(\mathbf{y})\ddot{s} + \hat{\mathbf{b}}(\mathbf{y})\dot{s}^2 + \hat{\mathbf{c}}(\mathbf{y}), \tag{6.6}$$

where

$$\begin{aligned}\hat{\mathbf{a}}(\mathbf{y}) &= \mathbf{M}(\mathbf{p}(s) + \mathbf{e})\mathbf{p}'(s), \\ \hat{\mathbf{b}}(\mathbf{y}) &= \mathbf{M}(\mathbf{p}(s) + \mathbf{e})\mathbf{p}''(s) + \mathbf{p}'(s)^\top \mathbf{C}(\mathbf{p}(s) + \mathbf{e})\mathbf{p}'(s), \\ \hat{\mathbf{c}}(\mathbf{y}) &= \mathbf{M}(\mathbf{p}(s) + \mathbf{e})[\mathbf{K}_p\mathbf{e} + \mathbf{K}_d\dot{\mathbf{e}}] \\ &\quad + 2\dot{\mathbf{e}}^\top \mathbf{C}(\mathbf{p}(s) + \mathbf{e})\mathbf{p}'(s)\dot{s} + \mathbf{h}(\mathbf{p}(s) + \mathbf{e}).\end{aligned}$$

We observe that if the tracking error is zero, the coefficients $\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}$ depend only on the path position s and not on the path velocity $\dot{s}$. Indeed in this case, the coefficients reduce to

$$\begin{aligned}\mathbf{a}(s) &= \mathbf{M}(\mathbf{p}(s))\mathbf{p}'(s), \\ \mathbf{b}(s) &= \mathbf{M}(\mathbf{p}(s))\mathbf{p}''(s) + \mathbf{p}'(s)^\top \mathbf{C}(\mathbf{p}(s))\mathbf{p}'(s), \\ \mathbf{c}(s) &= \mathbf{h}(\mathbf{p}(s)).\end{aligned}$$

We call $\mathbf{a}, \mathbf{b}, \mathbf{c}$ the nominal coefficients. Additionally, by inspection, we see that the coefficients $\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}$ are continuous with respect to the tracking error $\mathbf{e}, \dot{\mathbf{e}}$.

It follows from the definition of continuous functions that for any pair $(s, s^d) \in [0, 1] \times [0, \infty]$ and any positive number R there exists $r > 0$ such that for all $i \in [1 \dots n]$

$$\left\| \begin{bmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{bmatrix} \right\|_2 < r \implies \left\| \begin{bmatrix} \hat{a}_i(\mathbf{q}, \dot{\mathbf{q}}, s, s^d) - a_i(s) \\ \hat{b}_i(\mathbf{q}, \dot{\mathbf{q}}, s, s^d) - b_i(s) \\ \hat{c}_i(\mathbf{q}, \dot{\mathbf{q}}, s, s^d) - c_i(s) \end{bmatrix} \right\|_2 < R. \tag{6.7}$$

This result implies that for tracking errors with sufficiently small magnitude, the coefficients $\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}$ vary around the nominal coefficients $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Furthermore, since the

path velocity can always be assumed to be bounded, we can strengthen this result: there exists $\bar{r} > 0$ such that Eq. (6.7) holds for any pair (s, s^d).

6.2.3. Difficulties with designing path controllers

The fundamental difficulty with designing path controllers is that the coefficients $\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}$ are only available online. Hence, it is non-trivial to avoid situations in which there is no path acceleration that satisfies Eq. (6.6). A consequence of this infeasibility is that exponential convergence of the actual joint trajectory to the desired joint trajectory is not guaranteed because of saturating torque bounds. This difficulty also renders reaching the terminal set challenging.

6.3. Solving the Time-Optimal Path Tracking problem

6.3.1. Exponential stability with robust feasible control laws

In the last section, it was shown that if tracking errors have small magnitude, the coefficients of Eq. (6.6) vary around the nominal coefficients. Motivating by this observation, we introduce the notion of *robust feasible* control laws. Note that in this section, *control laws* refer to control laws for selecting path parameterization controls u, not the coupled control $[\boldsymbol{\tau}, u]$.

Specifically, a *control law* is a function that computes the path acceleration from the current path position, the current path velocity and the coefficients:

$$u(t) = \pi(s(t), \dot{s}(t), \hat{\mathbf{a}}(t), \hat{\mathbf{b}}(t), \hat{\mathbf{c}}(t)).$$

The control law π is *robust feasible* at the pair (s_0, s_0^d) if there exists $R > 0$ such that for any set of coefficients

$$\begin{aligned}
&\hat{\mathbf{a}}(t) := \mathbf{a}(s(t)) + \boldsymbol{\Delta}_a(t), \hat{\mathbf{b}}(t) := \mathbf{b}(s(t)) + \boldsymbol{\Delta}_b(t), \\
&\hat{\mathbf{c}}(t) := \mathbf{b}(s(t)) + \boldsymbol{\Delta}_c(t),
\end{aligned}$$

where the perturbations $\boldsymbol{\Delta}_a, \boldsymbol{\Delta}_b, \boldsymbol{\Delta}_c$ are arbitrary continuous functions satisfying

$$\|(\Delta_{a,i}(t), \Delta_{b,i}(t), \Delta_{c,i}(t))\|_2 < R, \forall t, i \in [1, \dots, n], \tag{6.8}$$

the generated parameterization $s(t)$ is feasible, that is

$$s(0) = s_0, \dot{s}(0) = s_0^d, \tag{6.9}$$
$$\exists T, s(T) = 1, \dot{s}(t) \in \mathbb{I}_{\text{end}}, \tag{6.10}$$
$$\forall t \in [0, T], \dot{s}(t) \geq 0 \tag{6.11}$$
$$\forall t \in [0, T], \text{Eq. (6.8) holds.} \tag{6.12}$$

The following result shows that a robust feasible control law ensures exponentially stable path tracking.

Proposition 4. *Consider a path tracking controller with (path acceleration) control law π. If π is robust feasible at (s_0, s_0^d) then the coupled system is exponentially stable at (s_0, s_0^d).*

Proof. Let R denote the bound on the magnitude of the perturbations such that π is robust feasible. Select $\bar{r} > 0$ such that Eq. (6.7) holds for R being the scalar bound in Eq. (6.19) and for all (s, s^d). Select $(\mathbf{q}_0, \mathbf{q}_0^d)$ such that the norm of the initial tracking error is less than $\bar{r}$.

Suppose we remove the torque bounds, the computed-torque tracking controller is exponentially stable. Thus, the tracking error converges exponentially to zero and its norm remains smaller than $\bar{r}$. Again using Eq. (6.7), it follows that the norm of the perturbations is always smaller than R.

Since π is robust feasible at (s_0, s_0^d) for R being the upper bound on the magnitude of the perturbations, the resulting parameterization $s(t)$ is feasible. It follows that the torque bounds are always satisfied. Therefore, the coupled path tracking system is exponentially stable at (s_0, s_0^d) according to the definition given in Section 6.2.1. □

Notice that the definition of robust feasible control laws does not require a specific R. In fact, as seen in the proof of Proposition 4, continuity of the coefficients guarantees the existence of $\bar{r}$ such that Eq. (6.8) holds for any value of R. It can be observed that $\bar{r}$ is the radius of a ball lying inside the region of attraction of the path tracking controller.

6.3.2. Characterizing robust feasible control laws

We now provide a characterization of robust feasible control laws. This development follows and extends the analysis of the Time-Optimal Path Parameterization problem in [96].

Discretize the interval $[0, 1]$ into $N + 1$ stages

$$0 =: s_0, s_1, \ldots, s_N := 1.$$

Define the *state* x_i and the *control* u_i as the squared velocity at s_i and the constant acceleration over $[s_i, s_{i+1}]$. One obtains the transition function

$$x_{i+1} = x_i + 2\Delta_i u_i, \tag{6.13}$$

where $\Delta_i := s_{i+1} - s_i$. We say that u_i "steers" x_i to x_{i+1}. See [96] for a derivation of Eq. (6.13).

At each stage, there are n pairs of torque bounds. The j-th pair of torque bounds at stage i is

$$\tau_{\min,j} \leq \tau_{ij} = \hat{a}_{ij} u_i + \hat{b}_{ij} x_i + \hat{c}_{ij} \leq \tau_{\max,j}. \tag{6.14}$$

The coefficients $\hat{a}_{ij}, \hat{b}_{ij}, \hat{c}_{ij}$ are assumed to vary around known nominal coefficients a_{ij}, b_{ij}, c_{ij}:

$$\hat{a}_{ij} = a_{ij} + \Delta_{a,ij},\ \hat{b}_{ij} = b_{ij} + \Delta_{b,ij},\ \hat{c}_{ij} = c_{ij} + \Delta_{c,ij}, \tag{6.15}$$

$$\|(\Delta_{a,ij}, \Delta_{b,ij}, \Delta_{c,ij})\|_2 \leq R. \tag{6.16}$$

The terms $(\Delta_{a,ij}, \Delta_{b,ij}, \Delta_{c,ij})$ are also called the perturbations. R is a parameter that be tuned to account for the magnitude of initial tracking errors. See the discussion at the end of Section 6.3.1 for more details. A control is feasible if it satisfies the constraints and is robust feasible if it satisfies all realizations of the constraints. Finally, the terminal velocity constraint is transformed to

$$x_N \in X_f := \{x : \sqrt{x} \in \mathbb{I}_{\text{end}}\}.$$

We say that a state is *robust controllable* at stage i if there exists a sequence of robust feasible controls that steers it to $\mathcal{X}_f$. The set of robust controllable states at stage i is called the *i-stage robust controllable set* $\mathcal{K}_i$.

In this discrete reformulation, the control law becomes a function π that maps the stage index, the state and the constraint coefficients $(i, x_i, \hat{\mathbf{a}}_i, \hat{\mathbf{b}}_i, \hat{\mathbf{c}}_i)$ to a control. Similarly, a control law is *robust feasible* at (i, x_i) if it steers x_i to the terminal set from stage i for any realization of the constraints with feasible controls.

We now give a characterization of robust feasible control laws.

Proposition 5. *For any state in $\mathcal{K}_i$ and any realization of the constraints, there exists at least one feasible control that steers that state to $\mathcal{K}_{i+1}$. If at any stage i, a control law steers states in $\mathcal{K}_i$ to $\mathcal{K}_{i+1}$, it is robust feasible at all robust controllable states at all stages.*

This characterization of robust feasible control laws is only useful if one can compute the robust controllable sets. To do so, we first introduce the notion of the robust

one-step set. Given a target set $\mathbb{I} \subseteq \mathbb{R}$, the i-stage *robust one-step set* $\mathcal{Q}_i(\mathbb{I})$ is the set of states such that at each state, there is a robust feasible control that steers it to $\mathbb{I}$.

Proposition 6. *The i-stage robust controllable sets, for $i \in [0, \dots, N]$, can be computed recursively by*

$$\mathcal{K}_N = X_f, \quad \mathcal{K}_i = \mathcal{Q}_i(\mathcal{K}_{i+1}). \tag{6.17}$$

A proof of this statement is omitted due to space constraints. Interested readers can refer to [96] for the proof of a similar result. We can now give a proof of Proposition 5 below.

Proof of Proposition 5. Let x_i be a state in $\mathcal{K}_i$. Since x_i is robust controllable, there exists a sequence of controls $(u_i, \dots, u_{N-1})$ that are robust feasible and the resulting sequence of states $(x_{i+1}, \dots, x_N)$ satisfies $x_N \in \mathcal{X}_N$. Observe that this implies x_{i+1} is robust controllable, and hence, u_i is a robust feasible control that steer x_i to $\mathcal{K}_{i+1}$.

Consider a control law that steers states in $\mathcal{K}_i$ for any stage i to $\mathcal{K}_{i+1}$. It is clear that this control law steers any robust controllable states to $\mathcal{K}_N$. Since $\mathcal{K}_N = \mathcal{X}_f$, see (6), the control law is robust feasible. □

A class of convex sets that can be handled quite efficiently is the class of *Conic-Quadratic* representable (CQr) sets [6]. A set of vectors ϵ is CQr if it is defined by finitely many conic-quadratic constraints

$$\left\| \mathbf{D}_i \begin{bmatrix} \epsilon \\ \nu \end{bmatrix} - \mathbf{d}_i \right\|_2 \leq \mathbf{p}_i^\top \begin{bmatrix} \epsilon \\ \nu \end{bmatrix} - q_i, \; i \in [1, \dots, k].$$

Proposition 7. *If $\mathbb{I}$ is an interval, the set of state and robust feasible control pairs (x, u) that satisfies $x + 2\Delta_i u \in \mathbb{I}$ is a CQr. Furthermore, $\mathcal{Q}_i(\mathbb{I})$ is an interval.*

Indeed, from Eq. (6.14) and Eq. (6.15), the j-th joint torque is given by

$$\tau_{ij} = \begin{bmatrix} a_{ij} & b_{ij} & c_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ x_i \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta_{a,ij} & \Delta_{b,ij} & \Delta_{c,ij} \end{bmatrix} \begin{bmatrix} u_i \\ x_i \\ 1 \end{bmatrix}.$$

Since the norm of the perturbation is bounded, see Eq. (6.16), one obtains the inequality

$$\tau_{ij} \leq \begin{bmatrix} a_{ij} & b_{ij} & c_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ x_i \\ 1 \end{bmatrix} + R \left\| \begin{bmatrix} u_i \\ x_i \\ 1 \end{bmatrix} \right\|_2. \tag{6.18}$$

It is clear that if and only if the right-hand side is not greater than $\tau_{\max,j}$, the pair (u_i, x_i) satisfies all realizations of this constraint. One obtains the conic-quadratic constraint

$$R \left\| \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_i \\ x_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\|_2 \leq - \begin{bmatrix} a_{ij} & b_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ x_i \end{bmatrix} - c_i + \tau_{\max,j}. \tag{6.19}$$

Note that the lower bound can be handled in a similar way. Instead of finding the upper bound of τ_{ij}, one derives the lower bound

$$\tau_{ij} \geq \begin{bmatrix} a_{ij} & b_{ij} & c_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ x_i \\ 1 \end{bmatrix} - R \left\| \begin{bmatrix} u_i \\ x_i \\ 1 \end{bmatrix} \right\|_2. \tag{6.20}$$

By requiring the right-hand side to be greater than or equal to $\tau_{\min,j}$, one obtains another conic-quadratic constraint.

Finally, if $\mathbb{I}$ is an interval, the constraint $x + 2\Delta_i u \in \mathbb{I}$ is equivalent to two linear inequalities, which are clearly CQr. $\mathcal{Q}_i(\mathbb{I})$ being an interval is a simple corollary.

From Proposition 7, one can formulate a pair of conic-quadratic optimization programs to compute the robust one-step set for any given target set. One program maximizes x while one program minimizes. Computing the robust controllable sets is then possible with Proposition 6.

6.3.3. A control law for time-optimal path tracking

Proposition 5 can be used to identify a class of control laws that are robust feasible, which, by Proposition 4 are exponentially stable. What is then the control law that realizes the shortest traversal time in this class? We give the following conjecture.

Conjecture 1. *In the class of control laws that for all i steers states in $\mathcal{K}_i$ to $\mathcal{K}_{i+1}$, the control law that always chooses* the greatest feasible controls *is time-optimal.*

We current do not have a proof of this conjecture. Regardless, in our experiments, the conjecture is verified by comparing the traversal time with the duration of the time-optimal path parameterization.

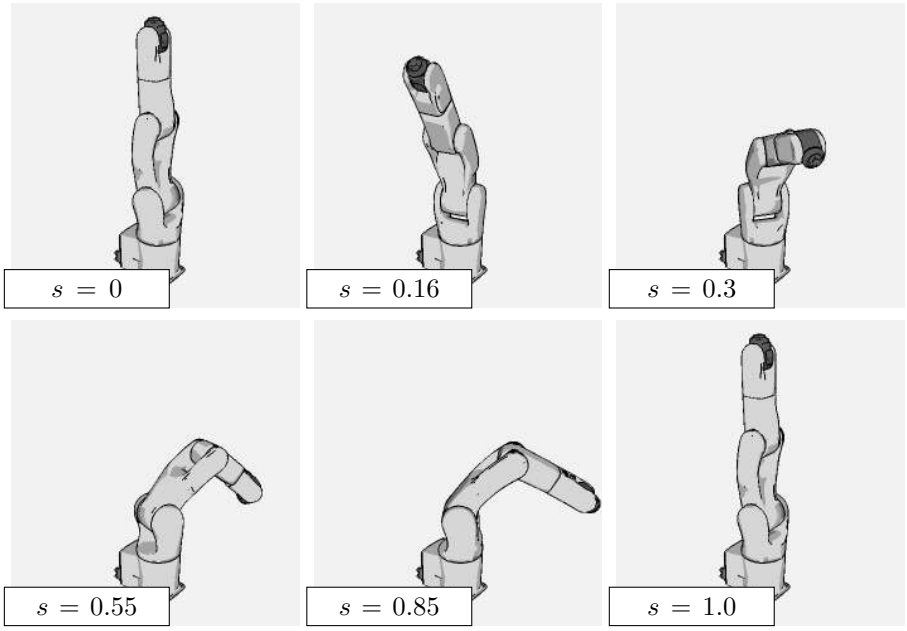


Figure 6.2.: The swinging motion used in the experiment.

6.4. Simulation results

We simulated a 6-axis robotic arm and controlled it to track a geometric path $\mathbf{p}(s)_{s\in[0,1]}$ with zero terminal velocity constraint. The torques bounds are

$$\boldsymbol{\tau}_{\max} = -\boldsymbol{\tau}_{\min} = [120., 280., 280., 120., 80., 80.](\mathrm{Nm}).$$

Fig. 6.2 visualizes the swinging motion. Initially the robot was at rest and had an initial joint positions error with magnitude $0.1\,\mathrm{rad}$. Forward dynamic computations were performed using OpenRAVE [35] and the `dopri5` solver. We sampled joint torques at sample time $1\,\mathrm{ms}$.

We implemented the time-optimal path tracking controller conjectured in Section 6.3.3, with the bounds on the norm of the perturbations R set uniformly to 0.5. The number of discretization step N was set to 100.

Computing the robust controllable sets excluding computations of the coefficients took $120\,\mathrm{ms}$ using our package toppra. We solved the conic-quadratic programs using the Python interface of `ECOS` [37]. Note that computing the coefficients involves evaluating the inverse dynamics twice per stage [99], which had a total running time of $40\,\mathrm{ms}$. Online computations of the controls $(\boldsymbol{\tau}, u)$ took $0.50\,\mathrm{ms}$ per time step. All computations were done on a single core of a laptop at $3.800\,\mathrm{GHz}$.

We compared our controller, called the Time-Optimal Path Tracking controller (TOPT), with the Online Scaling controller (OS) in [32] and the Computed-Torque Trajectory Tracking controller (TT) in [120]. The OS controller tracked the time-optimal path

parameterization of the given geometric path, while the TT controller tracked the time-optimal *trajectory*.

Table 6.1.: Tracking duration and max position errors

	TOPT	OS	TT
Max pos. err. (rad)	0.10	0.491	0.493
Tracking dur. (sec)	1.021	1.017	1.017

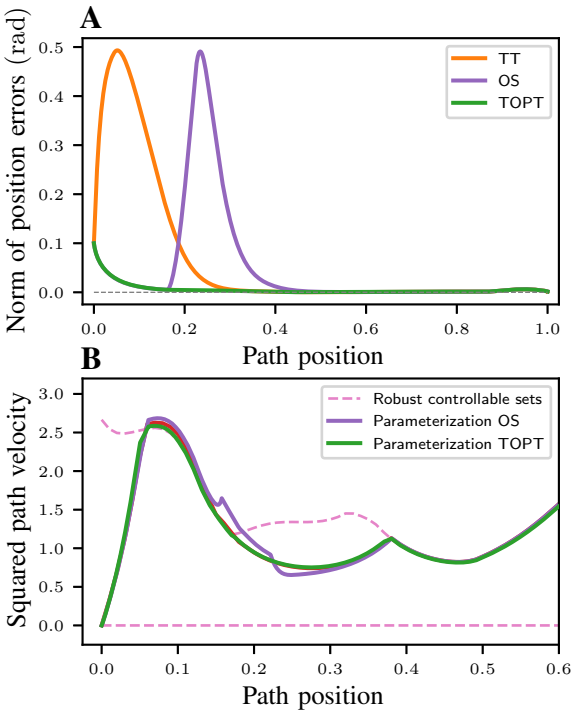


Figure 6.3.: **A**: Norms of joint position errors of three controllers: TOPT, OS and TT. **B**: The path position-squared path velocity space showing parameterizations (solid lines) generated online by the TOPT and OS controllers, the robust controllable sets (dashed lines).

We observe that the TT controller was incapable of handling the initial position error: Fig. 6.3A shows that position errors increased quickly reaching a maximum norm of $0.49\,\mathrm{rad}$ before stabilizing.

The OS controller was only able to regulate position errors during the initial segment. At $s \approx 0.18$, position errors increased sharply. See Fig. 6.3A. We note that at this instance, the OS controller was not able to find any *feasible* path acceleration. This event can be observed in Fig. 6.3B as a sharp spike on the generated parameterization.

The TOPT controller did not show any of the above problems. The joint positions converged quickly to zero. The total tracking duration of the TOPT controller was

slightly higher than the *optimal duration*: about 1% longer. See Table 6.1 for the durations.

Finally, we observed that the parameterization generated by the TOPT controller differed from the parameterization generated by the OS controller mostly *during decelerating path segments*. See for instance the path position interval $s \in [0.05, 0.15]$ in Fig. 6.3. Specifically, it can be seen that the TOPT controller "slowed down" in order to stay within the robust controllable sets. This helped the TOPT controller avoids the infeasibility at $s = 0.18$.

6.5. Summary

In this chapter, we considered the Time-Optimal Path Tracking problem: given a geometric path, find the control strategy to traverse the path time-optimally while regulating tracking errors. We have introduced the Time-Optimal Path Tracking controller and shown that the controller outperforms existing methods. The key innovation is the use of robust controllable sets, which intuitively define the sets of "safe" path parameterizations that can be tracked while accounting for possible variations of the coefficients. The technique used in this chapter is Reachability Analysis, a new method for analyzing path parameterization problems [96].

Several matters have been left for future investigations. Important questions include how to evaluate and optimize the region of attraction of path tracking controllers. Another direction is extending the approach to handle industrial manipulators with position or velocity interfaces and to account for higher-ordered constraints such as joint jerk bounds.

Chapter 7.

Conclusion

The recent recognition of robotics [60, 19, 48] has led to new challenges on *motion planning and control*. With the goal of resolving these challenges, this thesis has presented four contributions. The first is an efficient and robust algorithm that solves the Time-Optimal Path Parameterization (TOPP) problem for a wide class of robots and tasks. The second is an experimental case study of the challenging task of transporting objects with suction cup *critically fast*. As the third and fourth contributions, I have presented analyses and solutions for two well-known issues associated with time-optimal motions in practice: infinite joint jerk at switching points and poor tracking performance.

Looking back at the original TOPP problem formulated by Bobrow et al. [16]–computing the time-optimal path parameterization for robots subject to joint torque limits–it is not an overstatement to say that the problem has been solved and that there is little one could do to further improve the existing solutions. Sounding as audacious as it is, we can now solve TOPP for robots subject to, in addition to joint torque limits, *joint acceleration limits, joint velocity limits, Cartesian acceleration limits, Cartesian velocity limits, (frictional) contact stability constraints and Zero Moment Point constraint* in a robust and efficient manner. Furthermore, the TOPP algorithm presented in Chapter 3 has an expected computational complexity of $O(Nm)$, where N is the number of grid-points and m is the number of constraints. This computational complexity can be said to be optimal, because it is exactly the complexity of verifying whether a path parameterization satisfies all constraints [1].

With that said about the *original* TOPP problem, the *general* TOPP problem, which contains constraints other than those listed above, remains challenging. The algorithm presented in Chapter 5, as well as other Numerical Integration based algo-

[1] One needs to evaluate all m constraints at each grid-point and there are N grid-points.

rithms [38, 95], have edge cases that are difficult to solve robustly. Optimization based algorithms can produce locally optimal solutions robustly, but are slow. A fast and robust solution to this general TOPP problem, which has eluded researchers for years, might lie at a more fundamental level: analytical results on optimality of optimal control problems [74]. Considering the wealth of applied mathematical research literature on optimal control, it is hard to not suspect that the solution is lying somewhere, waiting to be discovered. Fortunately, the lack of an efficient solution for the general problem is not a major hurdle to a majority of robotic applications.

Even up until recently, computational robotic motion planning had yet been widely applied in the industries. One of the main reasons is the difficulty in computing path parameterizations that satisfy task constraints robustly and efficiently. I believe that the introduction of TOPP-RA–the TOPP algorithm in Chapter 3–has provided this missing piece and thereby enabled computational robotic motion planning in the industries. Indeed, we have seen how the challenging task of transporting objects with suction cup critically fast can be solved reliably in Chapter 4. The suction cup constraint, which guarantees that frictional and suction forces are within limits, has more than 3000 inequalities; and yet TOPP-RA consistently terminated in approximately 100 ms [2]. Our implementation of TOPP-RA on GitHub (https://github.com/hungpham2511/toppra) has also gained interests from robotic researchers developing commercial and research robotic applications.

Despite this welcoming achievement, research in motion planning and control for industrial robotics is far from lacking interesting open problems. Not considered in this thesis, motion planning and control for human-robot collaboration [83, 27] is one such problem. The most important requirement is to ensure that robots do not cause serious injuries to the human operators working alongside them. This requirement leads to several challenges. One group of challenges include modelling and predicting human operators' intentions and behaviors. Another is formulating appropriate safety constraints in order to maximize safety without affecting severely the optimality of the solutions. Principal solutions for these challenges would be beneficial even to fields extending far from industrial robotics where the ability to reason about optimal movements while maintaining high safety in the presence of human agents is greatly appreciated.

In recent years, research on robotic motion planning has been shifting toward nonlinear trajectory optimization and Deep Reinforcement Learning. The underlying trends motivating this shift are (i) the consideration of larger and more complex dynamic systems (e.g. legged robots, flexible robotics arms) and (ii) the availability of large quantities of data in the latter case. To fully utilize TOPP algorithms in consid-

[2] This implementation of TOPP-RA is in Python/Cython, which is not as efficient as C or C++.

eration of both trends, there are two important questions to address:

1. What is the role of TOPP algorithms in complex robotic systems that employ nonlinear trajectory optimization or Deep Reinforcement Learning?

2. How can we leverage available data from real executions to improve the quality of motion computed with TOPP algorithms?

Below I briefly discuss both questions.

As demonstrated in this thesis, motion planning based on TOPP is highly efficient and robust; however, it is only applicable to robots and tasks with first-order and second-order constraints. On the other hand, nonlinear trajectory optimization is applicable to a much larger class of constraints and objectives [11, 127, 107] (unfortunately with higher computational cost and lost of optimality guarantee). TOPP can provide complementary information useful to the trajectory optimizer:

1. One can use a TOPP algorithm to solve a simplified model of the task that has only first-order and second-order constraints. The resulting time-optimal trajectories (both minimum-time and maximum-time) can be used as initial guesses for the trajectory optimizer. In addition, the respective minimal and maximal trajectory durations can provide upper and lower bounds of achievable trajectories respectively.

2. Given the same computational budget, one can solve TOPP for a longer horizon than what is possible for nonlinear trajectory optimization. Hence, we can use a motion planner based on TOPP to plan *strategic* motions, taking advantage of the longer horizon, while employ nonlinear trajectory optimization to compute shorter but more accurate control input trajectories.

3. The set of *controllable* path velocities, which can be computed using TOPP-RA, defines a *controllable manifold* in the state-space. Any state on this manifold can be "controlled" to reach the desired goal state. Therefore, this manifold can be used as a terminal set for a trajectory optimizer to "guide" the system toward the desired goal state, which is possibly far in the future.

The recent advances in Deep Reinforcement Learning, or Deep Learning in general, have had great impacts on robotics. Researchers have demonstrated that Deep Reinforcement Learning algorithms can solve manipulation tasks end-to-end with no prior knowledge [71, 72], achieve large scale grasping with state-of-art performance [73] and perform dexterous in-hand manipulation [2]. These developments have shown clearly the advantages of Deep Reinforcement Learning. First, these algorithms are

applicable to very general situations, with almost no restriction on system dynamics nor constraints. Second, end-to-end algorithms have a straightforward pipeline for training and autonomously improving performance. Taking into consideration the cheaply available data from both real experiments and simulations, this is perhaps the strongest advantage of Deep Reinforcement Learning. Finally, Deep Reinforcement Learning algorithms are capable of seamlessly integrating motion planning, control and computer vision in a single place (i.e. a neural network), taking full advantage of the superiority of deep convolutional neural networks in understanding visual inputs [117, 56]. Achieving the same level of integration in non end-to-end robotic systems requires significant engineering effort. At this point, one is naturally inclined to ask: *What are the advantages of motion planning with TOPP comparing to Deep Reinforcement Learning algorithms, which apparently can account for a much higher level of complexity?*. In my opinion, motion planning based on TOPP has key advantages that are unlikely to be overcome in near future:

1. As discussed in the introduction of this thesis, robotic applications often have "hard" constraints, which if not satisfied cause failure. Motion planning systems that are based on TOPP can guarantee constraint satisfaction mathematically. On the other hand, for Deep Reinforcement Learning algorithms, it is much harder to achieve the same goal: the only option available to the learning algorithm developer is to collect more data and feed these data to the algorithm. How much data are sufficient to achieve the desired level of performance is often unknown.

2. Motion planners based on TOPP can perform very fast collision-free path planning using the RRT algorithm [69] or CHOMP [132]. On the other hand, there is no evidence that an end-to-end algorithm can perform fast collision-free motion planning.

3. Suppose that a collision-free path is already known, what about using Deep Reinforcement Learning for speed planning? As the matter of fact, Reinforcement Learning was proposed earlier [114] (referred to as Dynamic Programming) to solve TOPP. The resulting algorithm, however, requires far greater computational cost and achieves lower quality solution comparing to state-of-art TOPP algorithms. Moreover, researchers have recently suggested that for problems with known analytical solutions, Deep Reinforcement Learning algorithms can hardly beat analytical solutions in term of efficiency and robustness [105].

Nonetheless, the ability of learning-based methods to autonomously improve performance from data is desirable. We will next discuss how this property can be realized for a motion planning system based on TOPP.

In order to autonomously improve motion quality from data, the key ingredient is a module for estimating task and robot constraints. Indeed, recall that a TOPP algorithm receives as inputs the geometric path, a list of constraints and outputs the final trajectory. In most cases, the list of constraints is kept fixed, hence one can interpret TOPP as a computational layer with the constraints being the parameters. Motion quality–the likelihood of constraint violation and the level of sub-optimality–depends only on these parameters, which can be learned from data collected during execution. In particular, we can record trajectory execution results: joint measurements, object index, wrench measurements, grasping position and a Boolean output indicating whether the attempt is successful. Using these data, we can use techniques from Machine Learning to infer first-order and second-order constraints (the parameters) directly. These constraints can then be used to compute robot motions that are closer to the limits imposed by the task, which are executed to collect more data and obtain better estimations. Such a hybrid system can potentially leverage the best of both worlds to achieve (i) fast execution, robustness and guarantee of constraint satisfaction (from a TOPP algorithm) and (ii) realistic constraints inferred from data (using Machine Learning). With some engineering effort, it should not be difficult to automate the whole process, realizing an autonomously improving motion planning system.

In the above concluding remarks, I have argued that computational motion planning has sufficiently matured for industrial applications. An enabling factor is the introduction of fast and robust TOPP algorithms. The usefulness of these TOPP algorithms could extend far out of industrial robotics, with potential use cases even in highly complex systems that rely chiefly on nonlinear trajectory optimization for motion planning. Finally, leveraging TOPP algorithms with Machine Learning is an interesting direction that can further enhance the performance and streamline the development of future robots.

Appendix A.

Publications

The main Chapters 3, 4, 5, and 6 have been published as articles J2, C3, C1, and C2 respectively. Other articles are attached in this appendix after this page.

Journal papers

J1 Hung Pham and Quang-Cuong Pham. Robotic Manipulation of a Rotating Chain. *IEEE Transactions on Robotics*, Apr 2018.

J2 Hung Pham and Quang-Cuong Pham. A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. *IEEE Transactions on Robotics*, Jul 2018.

J3 Xu Zhang, Mingyang Li, Jian Hui Lim, Yiwei Weng, Yi Wei Daniel Tay, Hung Pham and Quang-Cuong Pham. Large-scale 3D printing by a team of mobile robots. *Automation in Construction*, Aug 2018.

Conference papers

C1 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)*, 2017.

C2 Hung Pham and Quang-Cuong Pham. Time-Optimal Path Tracking via Reachability Analysis. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

C3 Hung Pham and Quang-Cuong Pham. Critically fast pick-and-place with suction cups. Accepted at *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

C4 Hung Pham, Lim Jian Hui and Quang-Cuong Pham. Robotic 3D-printing for building and construction. In *Proceedings of the 2nd International Conference on Progress in Additive Manufacturing (Pro-AM 2016)*, 2016.

C5 Puttichai Lertkultanon, Jingyi Yang, Hung Pham, and Quang-Cuong Pham. Departure and Conflict Management in Multi-Robot Path Coordination. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Robotic Manipulation of a Rotating Chain

Hung Pham and Quang-Cuong Pham

***Abstract*—This paper considers the problem of manipulating a uniformly rotating chain: the chain is rotated at a constant angular speed around a fixed axis using a robotic manipulator. Manipulation is quasi-static in the sense that transitions are slow enough for the chain to be always in "rotational equilibrium." The curve traced by the chain in a rotating plane—its shape function—can be determined by a simple force analysis, yet it possesses a complex multisolutions behavior that is typical of nonlinear systems. We prove that the configuration space of the uniformly rotating chain is homeomorphic to a two-dimensional surface embedded in $\mathbb{R}^3$. Using that representation, we devise a manipulation strategy for transiting between different rotation modes in a stable and controlled manner. We demonstrate the strategy on a physical robotic arm manipulating a rotating chain. Finally, we discuss how the ideas developed here might find fruitful applications in the study of other flexible objects, such as in circular aerial manipulation with UAVs.**

***Index Terms*—Dynamics, path planning, shape control.**

I. Introduction

AN IDLE person with a chain in her hand will likely at some point start rotating it around a vertical axis, as in Fig. 1(A). After a while, he/she might be able to produce another mode of rotation, whereby the chain would curve inward, as in Fig. 1(B), instead of springing completely outward. With sufficient dexterity, he/she might even reach more complex rotation modes, such as in Fig. 1(C). Transitions into such complex rotation modes are, however, difficult to reproduce reliably as instabilities can quickly lead to unsustainable rotations [see Fig. 1(D)]. This paper investigates the mechanics of the transitions between different rotation modes and proposes a strategy to perform those transitions in a stable and controlled manner.

A. Context

There are several reasons why this problem is hard to solve. First, there are multiple solutions for a given control input (distance r between the attached end of the chain and the rotation axis, and angular speed ω). This ambiguity makes it difficult to devise a manipulation strategy directly in the control space. Second, some control inputs can quickly lead to "uncontrollable" behaviors of the chain, as illustrated in in Fig. 1(D).

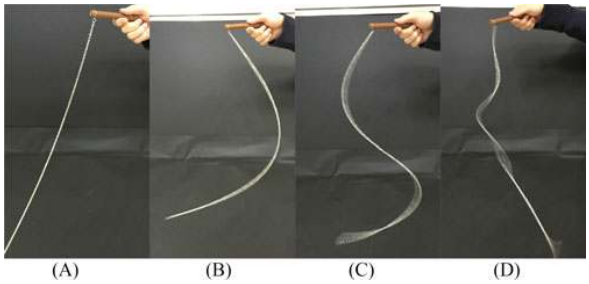


Fig. 1. Manual rotation of a chain around a vertical axis. A, B, and C: Uniform rotation modes 0, 1, and 2, respectively. D: Unstable behavior.

The theoretical study of the rotating chain and, in particular, of its rotation modes, has a long and rich history in the field of applied mathematics [1]–[7], which we will review in Section II-A. Here, by devising and implementing a *manipulation strategy* to stably transit between different rotation modes, we hope to provide a new, robotics-enabled, understanding of this problem. Indeed, at the core of our approach lie concepts specifically forged in the field of robotics, such as "configuration space," "stable configurations," "path-connectivity," etc.

As opposed to rigid bodies, flexible objects are generally characterized by an infinite number of degrees of freedom, which entails significant challenges when it comes to manipulation. Therefore, specific approaches have been developed in the field of robotics to study the manipulation of flexible objects, as reviewed in Section II-B.

The above studies are motivated by a number of practical applications. For the rotating chain in particular, applications include aerial manipulation by unmanned air vehicles (UAVs), which has recently received some attention, as discussed in more detail in Section II-C.

B. Contribution and Organization of the Paper

Our contribution in this paper is threefold. First, we study the case of arbitrary nonzero attachment radii r (see Section IV). This extends and generalizes existing works, which all focus on the case of zero attachment radius, and sets the stage for stable transitions between different rotation modes, which specifically require manipulating the attachment radius. In particular, we determine in that section the number of solutions to shape the equation for given values, i.e., r and ω.

Manuscript received March 23, 2017; revised September 4, 2017; accepted October 26, 2017. Date of publication December 21, 2017; date of current version February 5, 2018. This paper was recommended for publication by Associate Editor K. Hauser and Editor A. Billard upon evaluation of the reviewers' comments. This work was supported in part by Grant ATMRI:2014-R6-PHAM (awarded by the NTU and the Civil Aviation Authority of Singapore) and in part by the Medium-Sized Centre funding scheme (awarded by the National Research Foundation, Prime Minister's Office, Singapore). *(Corresponding author: Hung Pham.)*

The authors are with the Air Traffic Management Research Institute and Singapore Centre for 3D Printing, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798 (e-mail: pham0074@e.ntu.edu.sg; cuong.pham@normalesup.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2017.2775650

Second, we show that the configuration space of the uniformly rotating chain with variable attachment radius is homeomorphic to a two-dimensional surface embedded in $\mathbb{R}^3$ (see Section V). We study the subspace of stable configurations and establish that it is not possible to stably transit between rotation modes without going back to the low-amplitude regime.

Third, based on the above results, we propose a manipulation strategy for transiting between rotation modes in a stable and controlled manner (see Section VI). We show the strategy in action in a physical experiment where a robotic arm manipulates a rotating chain and makes it reliably transit between different rotation modes.

Before presenting our contribution, we review related works (see Section II) and recall Kolodner's equations of motion of the rotating chain (see Section III). Finally, we discuss possible applications and extensions and sketch some perspectives for future work (see Section VII).

II. Related Works

The manipulation of the rotating chain is relevant to a number of fields such as 1) applied mathematics, 2) flexible object manipulation in robotics, and 3) aerial manipulation. We now review the literature and describe the position of the current work with respect to each of those fields.

A. Theoretical Studies of the Rotating Chain

In applied mathematics, the study of the rotating chain was initiated in 1955 by a remarkable paper by Kolodner [1]. He established the existence of critical speeds $(\omega_i)_{i\in\mathbb{N}}$ such that there are no uniform rotations if $\omega < \omega_1$, and there are exactly n rotation modes for $\omega_n < \omega < \omega_{n+1}$. In [2], Caughey studied the rotating chain with small but nonzero attachment radii. The results obtained by Caughey extend Kolodner's and agree with our study of the low-amplitude regime. In [3], Caughey investigated the rotating chain with both ends attached. In [5], Stuart considered the original rotating chain problem using bifurcation theory, and arrived at the same results as Kolodner. In [4], Wu considered the large angular speeds regime. In [7], Toland initiated a new approach based on the calculus of variation, but did not obtain new significant results, as compared to Kolodner.

The common point of all previous works is that the chain is attached to the rotation axis, or very close to it [2]. Yet, reliably observing and transiting between different rotation modes precisely require using arbitrary nonzero attachment radii r, the distance between the attached end, and the rotation axis. The current paper extends previous studies by specifically considering arbitrary attachment radii.

B. Robotic Manipulation of Flexible Objects

Within the field of robotics, the manipulation of flexible objects is studied along two main directions. A first direction is topological: one is mainly interested in the order and sequence of the manipulation rather than in the precise behavior of the flexible object. Examples include origami folding [8], laundry folding [9], or rope-knotting [10], [11].

The second research direction is concerned with the precise shape and dynamics of the manipulated object. Within this research direction, one can distinguish two main approaches. The first approach discretizes the flexible object into a large number of small rigid elements, and subsequently carries out finite-element calculations, see, e.g., [6], [12], and [13] for inextensible cables or [14] and [15] for concentric tube robots. This approach can be applied to any type of flexible objects as long as a dynamical model is available. However, it usually yields no *qualitative* understanding of the manipulation. For example, while finite-element calculations can compute the shape of the rotating chain for various control inputs, they can establish neither the existence of different rotation modes, nor the manipulation strategies to transit between different modes.

By contrast, the second approach considers the flexible object as the solution of a (partial) differential equation and tries to establish qualitative properties of this solution. While this approach is harder to put in place—usually because of the complex mathematical calculations and concepts involved—it can lead to stunning and insightful results. For example, Bretl and McCarthy [16] and Bretl and Borum [17] established that the configuration space of the Kirchhoff elastic rod is of dimension 6 and that it is path-connected. Such results would be impossible to obtain via finite-element methods.

The present study of the rotating chain is inscribed within this analytical approach. From the dynamic model of the rotating chain, we investigate qualitative properties of its configuration space: dimension, connectivity, and stability. These properties are in turn crucial to devise a manipulation strategy to stably transit between different rotation modes.

C. Aerial Manipulation

Although the study of the rotating chain first stemmed out of scientific curiosity, it has recently found applications in aerial manipulation. In [13] and [18], Murray and Williams and Trivailo considered a fixed-wing aircraft towing a long cable whose other end is free. The circular flying pattern imprints a pseudo-stationary shape to the cable, which in turn allows precisely controlling the position of the free end. Practical applications of this scheme include remote sensing in isolated areas [18], payload delivery and pickup [12], [18], [19], mobile 3-D printing [20] or more recently, recovery of micro air vehicles [21]–[23]. In the latter application, the micro vehicles are able to attach themselves to the towed end, which moves at a relatively slower speed than that of the aircraft. The recent surge of interest in UAVs also offers many potential applications: Merz and Johansen [19] studied a single UAV flying circularly while towing a cable, Sreenath *et al.* [24] deals with general (noncircular) aerial manipulation, while Michael *et al.* [25] targets cooperative manipulation using a team of UAVs.

The above works are based on dynamic simulation [6], [12], [26], or numerical optimal control [23], [27]. Physical experiments were found to agree with simulations [18]. However, there are a number of questions these works are unable to address, for instance: 1) under which conditions are there multiple solutions to the same set of controls (fly radius and angular

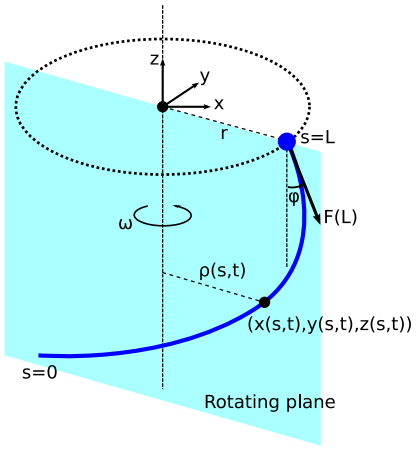


Fig. 2. Chain rotating around a fixed vertical axis. At a time instant t, the chain describes a 3-D curve parameterized by s: $s = 0$ at the free end, $s = L$ at the attached end, where L is the length of the chain.

speed)? 2) how to avoid or initiate "jumps" between different quasi-static rotational solutions [6]? Here, we precisely answer these questions for the case of a simple rotating chain, without considering aerodynamic drag or end mass. We also discuss how the method can be extended to include these effects, offering thereby solid theoretical foundations for developing safe and stable applications in circular aerial manipulation.

III. Background and Problem Setting

A. Equations of Motion of the Rotating Chain

Here we recall the main equations governing the motion of the rotating chain initially obtained by Kolodner [1]. Fig. 2 depicts an inextensible and homogeneous chain of length L and linear density μ that rotates around a vertical Z-axis. One end of the chain is maintained at the attachment radius r from the rotation axis, while the other end is free. Note that the case of a chain with tip mass can be reduced to this case, see Appendix A.

Let $\mathbf{x}(s,t) := [x(s,t), y(s,t), z(s,t)]^\top \in \mathbb{R}^3$ denote a length -time parameterization of the chain where s equals zero at the free end and equals L at the attached end (see Fig. 2). Next, let $F(s,t) \geq 0$ be the tension of the chain. Neglecting aerodynamic effect, one writes the equation of motion for the chain as

$$\mu \ddot{\mathbf{x}} = (F\mathbf{x}')' + \mu \mathbf{g} \tag{1}$$

where $\dot{\square}$ and $\square'$ denote differentiation with respect to t and s, respectively; $\mathbf{g} := [0, 0, -g]^\top$ is the gravitational acceleration vector. The inextensibility constraint can be written as

$$\|\mathbf{x}(s,t)'\|_2 = 1. \tag{2}$$

We seek solutions that are *uniform rotations*: those which have constant shape in a plane that rotates around the Z-axis. In this case, the motion of the chain becomes

$$\begin{aligned} x(s,t) &= \rho(s)\cos(\omega t), \\ y(s,t) &= \rho(s)\sin(\omega t), \\ z(s,t) &= z(s) \end{aligned} \tag{3}$$

where the function $\rho(s)$ is called the *shape function* of the chain. Directly from the inextensibility constraint (2), one has

$$\|(\rho(s), z(s))\|_2 = \sqrt{\rho'(s)^2 + z'(s)^2} = 1. \tag{4}$$

From the uniform rotation assumption, the tension of the chain $F(s,t)$ is in fact time independent.

Substituting the above expressions into (1) yields

$$(F\rho')' + \mu\rho\omega^2 = 0, \tag{5}$$

$$(Fz')' - \mu g = 0 \tag{6}$$

where F, ρ, z are functions of s. Integrating (6) and noting that the tension at the free end vanishes (i.e., $F(0) = 0$) yield

$$Fz' = \int_0^s \mu g \, \mathrm{d}\lambda = \mu g s. \tag{7}$$

Next, by the inextensibility constraint (4), one has

$$F = \frac{\mu g s}{z'} = \frac{\mu g s}{\sqrt{1 - \rho'^2}}. \tag{8}$$

Substituting (8) into (5) yields the governing equation for the shape function $\rho(s)$

$$\frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{\mu g s}{\sqrt{1-\rho'^2}}\rho'\right) + \mu\rho\omega^2 = 0 \tag{9}$$

subject to the following boundary condition:

$$\rho(L) = r. \tag{10}$$

Remark that we have applied two boundary conditions:) tension at the free end must be zero: $F(0) = 0$; and 2) $\mathbf{x}(L,t)$ equals the reference trajectory traced by the robotic manipulator (or the aircraft trajectory in the towing problem).

B. Problem Formulation

We can now define the *configurations* and the *control inputs* of a rotating chain.

Definition 1: (Configuration) A configuration of the rotating chain is a pair $q := (\omega, \rho)$, where $\omega \geq 0$ is a rotation speed and ρ is a shape function satisfying the governing equation (9) and that $\rho(0) \geq 0$. The set of all such configurations is called the configuration space of the rotating chain and denoted as s$\mathcal{C}$.

Definition 2: (Control input) A control input is a pair (r, ω), where $r \geq 0$ is an attachment radius and $\omega \geq 0$ is a rotation speed. The set of all inputs is called the control space and denoted as $\mathcal{V}$. If (9) has nontrivial solutions with boundary conditions and parameters defined by the input (r, ω) then the input is called *admissible*.

Note that the condition $\rho(0) \geq 0$ in Definition 1 precludes duplicate solutions. Any configuration (ω, ρ) corresponds to two possible solutions: one has shape function ρ, while the other has shape function $-\rho$, both rotating at angular speed ω. The later

solution can be obtained by rotating the former solution by 180°. A similar remark applies to the definition of the control space $\mathcal{V}$ where we require positive attachment radii.

We can formulate the chain manipulation problem as follows: given a pair of starting goal configurations $(q_{\text{init}}, q_{\text{goal}})$ find a control trajectory $(0,1) \to \mathcal{V}$ that brings the chain from q_{init} to q_{goal} without going through instabilities (instabilities will be discussed in Section V-C).

IV. Forward Kinematics of the Rotating Chain With Non-Zero Attachment Radius

A. Dimensionless Shape Equation

Still following Kolodner, we convert (9) into a dimensionless equation, more appropriate for subsequent analyses. Consider the changes of variable

$$u := \frac{\rho'}{\sqrt{1-\rho'^2}} \frac{s\omega^2}{g}, \quad \bar{s} := \frac{s\omega^2}{g} \tag{11}$$

which, by combining with (9), leads to

$$\frac{\mathrm{d}u}{\mathrm{d}\bar{s}} + \rho \frac{\omega^2}{g} = 0. \tag{12}$$

One can now differentiate (12) with respect to $\bar{s}$ to arrive at

$$\frac{\mathrm{d}^2}{\mathrm{d}\bar{s}^2} u + \rho' = 0$$

which is combined with the relation

$$\rho' = \frac{u}{\sqrt{\bar{s}^2 + u^2}} \tag{13}$$

to yield the dimensionless differential equation

$$\frac{\mathrm{d}^2}{\mathrm{d}\bar{s}^2} u(\bar{s}) + \frac{u(\bar{s})}{\sqrt{\bar{s}^2 + u(\bar{s})^2}} = 0. \tag{14}$$

We first consider the boundary condition at $\bar{s} = 0$. By definition of u, one has $u(0) = 0$. The end boundary condition $\rho(L) = r$ implies that

$$u'\left(L\frac{\omega^2}{g}\right) = -r\frac{\omega^2}{g} \tag{15}$$

where $\square'$ denotes in this context differentiation with respect to $\bar{s}$.

We summarize the boundary conditions on u as

$$u(0) = 0, \quad u'(\bar{L}) = \bar{r} \tag{16}$$

where

$$\bar{L} := L\omega^2/g, \quad \bar{r} := -r\omega^2/g. \tag{17}$$

This is the standard form of a boundary value problem (BVP).

Remark: The distance from the free end to the Z-axis is denoted by ρ_0 . Using (12), one has

$$u'(0) = a \tag{18}$$

where $a = -\rho_0\omega^2/g$.

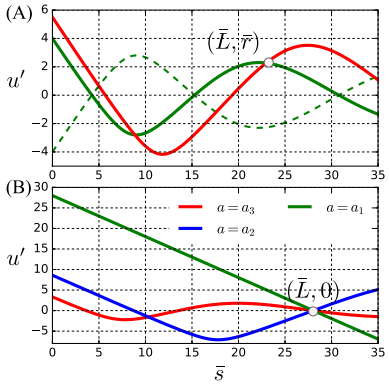


Fig. 3. A: Shooting from different initial guesses of $u'(0) = a$. There might be more than one initial value (green and red) that satisfy the end condition $u'(\bar{L}) = \bar{r}$. B: $a_1, a_2, \ldots$ are different initial values of $u'(0)$ that yield $u'(\bar{L}) = 0$; a_i denotes the initial guess such that the ith zero of u' coincides with $\bar{L}$

Remark: Applying L'Hôpital rule twice, one finds that

$$\lim_{\bar{s}\to 0} \frac{u(\bar{s})}{\sqrt{\bar{s}^2 + u(\bar{s})^2}} = \frac{a}{\sqrt{1+a^2}}.$$

Thus, the differential equation (14) is well-defined at $\bar{s} = 0$.

B. Shooting Method

We numerically solve the BVP posed in the last section using the *simple shooting method* [28]. Given a control input (r, ω), the method finds resulting configurations as follows:

1) compute $(\bar{r}, \bar{L})$ from (r, ω) using (17);
2) repeat until convergence:
   a) guess an initial value $a \in \mathbb{R}$ for $u'(0)$ or use the value from the last iteration;
   b) integrate (14) from the initial condition $(u(0), u'(0)) = (0, a)$ at $\bar{s} = 0$ to $\bar{s} = \bar{L}$;
   c) check whether $u'(\bar{L}) = \bar{r}$;
   d) if not, refine the guess a by, e.g., Newton's method;
3) recover $\rho(s)$ from $u'_{\text{last\_iter}}(\bar{s})$.

One can then recover $z(s)$ using $\rho(s)$, the inextensibility constraint (4), the boundary condition $z(L) = 0$ and the fact that $z'(s) \geq 0$ [see, (7)]. Also, for a given tuple $(\bar{r}, \bar{L})$, there might be multiple solutions to the BVP which translates into multiple configurations for a given control input [see Fig. 3(A)].

Remark: It is straightforward to see that if $u(\bar{s})_{\bar{s}\in[0,\bar{L}]}$ is a solution of (14), then $-u(\bar{s})_{\bar{s}\in[0,\bar{L}]}$ is also a solution. Therefore, there is no loss of generality to consider only non-negative values of a, as integrating from $-a$ leads to the same configuration.

C. Number of Configuration

We now analyze the number of solutions for different parameters. The function $u'(\bar{s})$ obtained by integrating from $(u(0), u'(0)) = (0, a)$ is denoted by $u'_a(\bar{s})$. Following Kolodner, let $\zeta_i(a)$ be the ith zero of $u'_a(\bar{s})$. The function $\zeta_i(a)$ has the following properties ([1, Th. 2]):

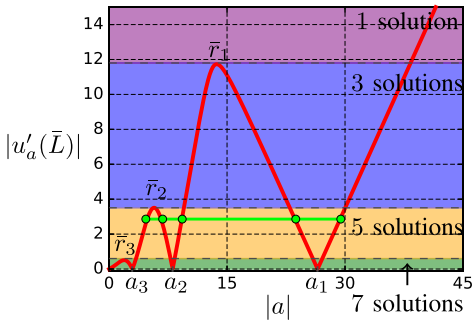


Fig. 4. Graph of $|u'_a(\bar{L})|$ versus $|a|$. The main text shows that if $\bar{r}_{i+1} < |\bar{r}| < \bar{r}_i$, then there are $2i+1$ nontrivial solutions. The green line illustrates the case $\bar{r}_3 < |\bar{r}| < \bar{r}_2$ where there are five nontrivial solutions (green disks).

1) $\zeta_i(a)$ is well-defined for all $i \in \mathbb{N}$ and is a strictly increasing function of a over $(0, +\infty)$;
2) $\lim_{a\to 0} \zeta_i(a) = h_i^2/4 =: \lambda_i$ where h_i is the ith zero of the Bessel function J_0 (Appendix B);
3) $\lim_{a\to+\infty} \zeta_i(a) = +\infty$.

Next, let us define a_i as the absolute value of a such that $\zeta_i(a) = \bar{L}$, i.e.,

$$a_i := |\zeta_i^{-1}(\bar{L})|. \tag{19}$$

By the properties of ζ_i, a_i exists if and only if $\lambda_i \le \bar{L}$ and when it exists, it is unique since $\zeta_i(a)$ is a strictly increasing function of a.

Fig. 3(B) shows the construction of a_1, a_2, and a_3. One can also observe that the a_i's form a decreasing sequence, i.e.,

$$a_1 > a_2 > a_3 > \ldots > a_n$$

where n is the largest i so that $\lambda_i \le \bar{L}$.

We now turn to the general case when $\bar{r}$ is not necessarily zero. For a given value of $(\bar{r}, \bar{L})$, it can be seen that the number of configurations equals the number of intersections that the graph of $u'_a(\bar{L})$ versus a makes with the horizontal lines $u'_a(\bar{L}) = \bar{r}$ and $u'_a(\bar{L}) = -\bar{r}$.

Further, since $\bar{r}$ and $-\bar{r}$ refer to the same radius and that $\rho(s)$ and $-\rho(s)$ refer to the same shape function, the number of intersections the graph of $|\bar{\rho}_a(\bar{L})|$ versus $|a|$ makes with the horizontal line $|u'_a(\bar{L})| = |\bar{r}|$ equals the number of configurations (see Fig. 4).

By inspecting Fig. 4, $|u'_a(\bar{L})|$ is zero at a_i and a_{i+1}; note moreover that $|u'_a(\bar{L})|$ increases as a increases from a_{i+1}, reaches a maximum at some a_i^*, and then decreases as a increases from a_i^* to a_i.[1] Let us denote the maximum reached by $|u'_a(\bar{L})|$ between a_i and a_{i+1} by $\bar{r}_i$, i.e.,

$$\bar{r}_i := |u'_{a_i^*}(\bar{L})| = \max_{a_{i+1}<a<a_i} |u'_a(\bar{L})|, \quad \text{for } i < n; \tag{20}$$

$$\bar{r}_n := \max_{0<a<a_n} |u'_a(\bar{L})|. \tag{21}$$

[1]This claim is based on numerical observations.

One can next observe that the $\bar{r}_i$'s form a decreasing sequence,[2] i.e.,

$$\bar{r}_1 > \bar{r}_2 > \bar{r}_3 > \ldots > \bar{r}_n.$$

We can now state the following proposition, whose proof results directly from the examination of Fig. 4.

Proposition 1: Let n be the largest i so that $\lambda_i \le \bar{L}$. The number of nontrivial configurations of a uniformly rotating chain depends on $|\bar{r}|$ as follows:

1) if $|\bar{r}| = 0$, there are n nontrivial solutions;
2) if $0 < |\bar{r}| < \bar{r}_n$, there are $2n+1$ nontrivial solutions;
3) if $\bar{r}_{i+1} < |\bar{r}| < \bar{r}_i$ for $i \in [1, n-1]$, there are $2i+1$ nontrivial solutions;
4) if $|\bar{r}| = \bar{r}_i$ for $i = [1, n]$, there are $2i$ nontrivial solutions;
5) if $|\bar{r}| > \bar{r}_1$, there is one nontrivial solution.

D. Rotation Modes

By the change of variable (11), $u' = \rho\omega^2/g$, the number of zeros of $u'(\bar{s})_{\bar{s}\in(0,\bar{L})}$, corresponds to the number of times the chain crosses the rotation axis. We can now give an operational definition of rotation modes.

Definition 3 (Rotation modes): A chain is said to be rotating in mode i if its shape crosses the axis exactly i times or, in other words, if the function $u'(\bar{s})_{\bar{s}\in(0,\bar{L})}$ has exactly i zeros.

Let us reinterpret Proposition 1 in terms of rotation modes. Consider a positive $\bar{r}$ verifying $\bar{r}_{i+1} < \bar{r} < \bar{r}_i$. In Fig. 4, the horizontal line $|u'(\bar{L})| = \bar{r}$ intersects the graph of $|\bar{\rho}_a(\bar{L})|$ versus $|a|$ at $2i+1$ points. Call the X-coordinates of these points $b_1 > b_2 > \cdots > b_{2i+1}$. Remark that

1) $b_1 > a_1$, thus by definition of a_1, the function $u'_{b_1}(\bar{s})$ has no zero in $(0, \bar{L})$, i.e., the chain rotates in mode 0;
2) $a_1 > b_2 > b_3 > a_2$, thus by definition of a_1, a_2, the functions $u'_{b_2}(\bar{s})$ and $u'_{b_3}(\bar{s})$ have each one zero in $(0, \bar{L})$, i.e., the chain rotates in mode 1;
3) more generally, for any $k \in [1, i]$, $a_{k-1} > b_{2k} > b_{2k+1} > a_k$, thus by definition of a_{k-1}, a_k, the functions $u'_{b_{2k}}(\bar{s})$ and $u'_{b_{2k+1}}(\bar{s})$ have each k zeros in $(0, \bar{L})$, i.e., the chain rotates in mode k.

Fig. 5 illustrates the above discussion for $i = 2$.

V. Analysis of the Configuration Space of the Rotating Chain

In the previous section, we have established a relationship between the control inputs and the configurations. Here, we investigate the properties of the configuration space and of the subspaces of stable configurations. In particular, a crucial question for manipulation, which we address, is whether the stable subspace is *connected*, allowing for stable and controlled transitions between different modes.

A. Parameterization of the Configuration Space

From now on, we make two technical assumptions: 1) the distance $\rho(0)$ from the free end of the chain to the rotation axis

[2]We have not yet been able to prove rigorously that the sequence is indeed decreasing.

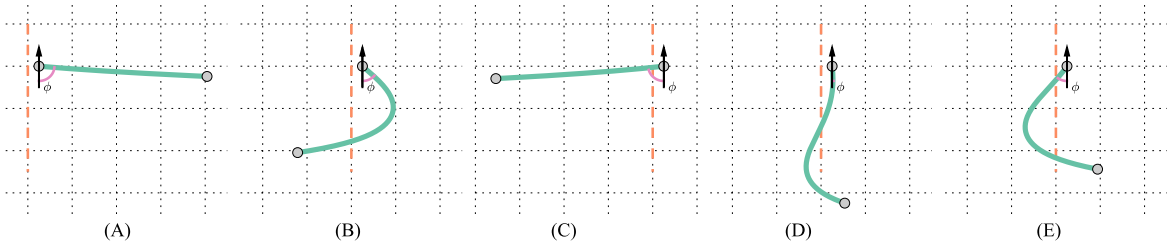


Fig. 5. Rotation modes for $\bar{r}$ where $\bar{r}_3 < \bar{r} < \bar{r}_2$ ($i = 2$). According to Proposition 1, there are $2i + 1 = 5$ solutions, depicted in A–E. A rotation is said to be in mode i if the chain shape crosses the rotation axis (dashed line) i times. A: solution in mode 0, corresponding to b_1 (for the explanation of the numbers b_i, see main text). B, C: solutions in mode 1, corresponding to b_2, b_3. D, E: solutions in mode 2, corresponding to b_4, b_5. In addition, the analysis of Section V-C shows that A, B, and D are stable, while C and E are unstable.

is upper-bounded by some $\rho_{\max}$; and 2) the rotation speed ω is upper-bounded by some $\omega_{\max}$. Note that these two assumptions do not reduce the generality of our formulation since they simply assert that there exist some finite bounds, which could be arbitrarily large. From (17) and (18), the two assumptions next imply that a and $\bar{L}$ are upper-bounded by some constants $a_{\max}$ and $\bar{L}_{\max}$. We can now prove a first characterization of the configuration space.

Proposition 2 (and definition): Define the parameter space $\mathcal{A}$ by

$$\mathcal{A} := (0, a_{\max}) \times (0, \bar{L}_{\max}).$$

There exists a homeomorphism $f : \mathcal{A} \to \mathcal{C}$.

This proposition implies that, despite 1) the potentially infinite dimension of the space of all shape functions ρ and 2) the one-to-many mapping between control inputs and configurations, the configuration space of the rotating chain is actually of dimension 2 and has a very simple structure: $\mathcal{A}$ is simply a 2D box.

The first dimension, a, is proportional to the distance of the free end to the rotation axis. Thus, choosing the free end rather than the attached end as reference point allows finding a one-to-one mapping with the shape function. The second dimension, $\bar{L}$, is defined by $\bar{L} := L\omega^2/g$. Since the length L of the chain is fixed, $\bar{L}$ changes as a function of the angular speed ω.

To simplify the notations, we define $\mathbf{u} := (u, u')$ and rewrite (14) as a dimensionless ODE

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\bar{s}} = \mathbf{X}(\mathbf{u}, \bar{s}). \tag{22}$$

We can now give a proof for Proposition 2.

Proof: The mapping f is essentially the shooting method described in Section IV-B. Given a pair $(a, \bar{L}) \in \mathcal{A}$, we first obtain ω from $\bar{L}$ using the relationship $\bar{L} = L\omega^2/g$. Next, we integrate the ODE (22) from the initial condition

$$\mathbf{u}(0) = (0, a)$$

until $\bar{s} = \bar{L}$ to obtain $u'(\bar{s})$ for $\bar{s} \in (0, \bar{L})$. Finally, we obtain ρ from u' using (11).

1) Surjectivity of f. Let $(\omega, \rho) \in \mathcal{C}$. Since ρ verifies (9), one can perform the change of variables (11) and obtain u and u'. Next, consider $a = u'(0)$ and $\bar{L} = L\omega^2/g$. One has clearly $a \in (0, a_{\max})$, $\bar{L} \in (0, \bar{L}_{\max})$, and $f((a, \bar{L})) = (\omega, \rho)$.
2) Injectivity of f. Assume that there are $(a_1, \bar{L}_1) \neq (a_2, \bar{L}_2)$ such that $f(a_1, \bar{L}_1) = f(a_2, \bar{L}_2) = (\omega, \rho)$. One has $a_1 = a_2 = -\rho(0)\omega^2/g$ and $\bar{L}_1 = \bar{L}_2 = L\omega^2/g$, which implies the injectivity.
3) Continuity of f. We show in the Appendix C that the ODE (22) is Lipschitz. It follows that the function $u'(\bar{s})$ for $0 \leq \bar{s} \leq \bar{L}$ depends continuously on its initial condition, which implies that $\rho(s)$ depends continuously on a.
4) Continuity of f^{-1}. It can be seen from the injectivity proof that a and $\bar{L}$ depend continuously on ω and $\rho(0)$, and the latter depends in turn continuously on ρ.

■

Next, we establish a homeomorphism between the parameter space and a smooth surface in 3-D, which allows an intuitive visualization of the configuration space.

Proposition 3 (and definition): For a given $a \in (0, a_{\max})$, integrate the differential equation (22) from $(0, a)$ until $\bar{s} = \bar{L}_{\max}$. The set $(\bar{s}, u(\bar{s}), u'(\bar{s}))_{\bar{s} \in (0, \bar{L}_{\max})}$ is then a 1-D curve in $\mathbb{R}^3$. The collection of those curves for a varying in $(0, a_{\max})$ is a 2-D surface in $\mathbb{R}^3$, which we denote by $\mathcal{S}$ (see Fig. 6).

There exists a homeomorphism $l : \mathcal{A} \to \mathcal{S}$.

Proof: The construction of l follows from the definition: given a pair $(a, \bar{L}) \in \mathcal{A}$, integrate (22) from $(0, a)$ until $\bar{s} = \bar{L}$. Then define $l(a, \bar{L}) := (\bar{L}, \mathbf{u}(\bar{L}))$.

1) Surjectivity of l. Consider a point $(\bar{L}, \mathbf{u}) \in \mathcal{S}$. By definition of $\mathcal{S}$, there exists $a \in (0, a_{\max})$ so that integrating (22) from $(0, a)$ reaches $\mathbf{u}$ at $\bar{s} = \bar{L}$. Clearly, $l(a, \bar{L}) = (\bar{L}, \mathbf{u})$.
2) Injectivity of l. This results from the Uniqueness theorem for ODEs, see Appendix C.
3) Continuity of l. From the Continuity theorem for ODEs (Appendix C), it is clear that the end point $(\bar{L}, \mathbf{u}(\bar{L})) \in \mathcal{S}$ depends continuously on the initial condition a.
4) Continuity of l^{-1}. Consider two points $(\bar{L}_1, \mathbf{u}_1^*)$, $(\bar{L}_2, \mathbf{u}_2^*) \in \mathcal{S}$ that are sufficiently close to each other, i.e.,

$$|\bar{L}_1 - \bar{L}_2| \leq \delta, \quad \|\mathbf{u}_1^* - \mathbf{u}_2^*\| \leq \delta,$$

for some δ that we shall choose later. Consider the curves $\mathbf{u}_1, \mathbf{u}_2$ such that $\mathbf{u}_1(\bar{L}_1) = \mathbf{u}_1^*$ and $\mathbf{u}_2(\bar{L}_2) = \mathbf{u}_2^*$. By the Continuity theorem (Appendix C) one has, for some

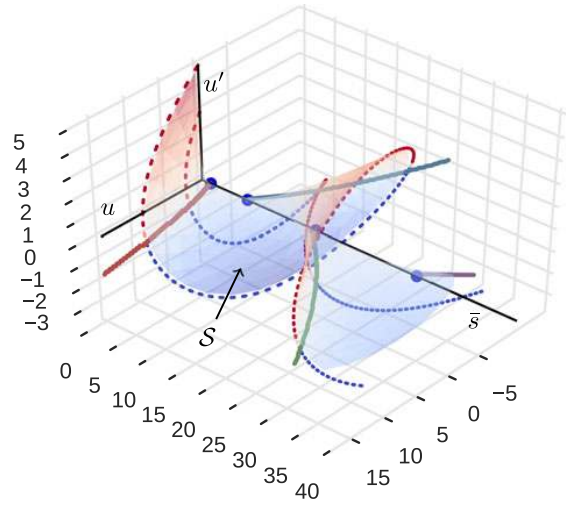


Fig. 6. Surface $\mathcal{S}$ that is homeomorphic to the configuration space $\mathcal{C}$. We depict two solution curves on the surface $\mathcal{S}$ (dashed lines), integrated from two different values of a (large and medium). Red, blue, green, and purple lines represent, respectively, the first, second, third, and fourth zero-radius loci (see Proposition 4).

appropriate constant M

$$\begin{aligned}\|\mathbf{u}_1(0)-\mathbf{u}_2(0)\| &\le e^{M\bar{L}_1}\|\mathbf{u}_1(\bar{L}_1)-\mathbf{u}_2(\bar{L}_1)\| \\ &\le e^{M\bar{L}_1}\left(\|\mathbf{u}_1(\bar{L}_1)-\mathbf{u}_2(\bar{L}_2))\|+\|\mathbf{u}_2(\bar{L}_2)-\mathbf{u}_2(\bar{L}_1)\|\right) \\ &\le e^{M\bar{L}_1}(\delta+M|\bar{L}_1-\bar{L}_2|)=e^{M\bar{L}_1}(M+1)\delta,\end{aligned}$$

where the last inequality come from the uniform boundedness of $\mathbf{u}$. For any ϵ, it suffices therefore to choose $\delta := \frac{\epsilon e^{-M\bar{L}_1}}{M+1}$ so that $|a_1-a_2| = \|\mathbf{u}_1(0)-\mathbf{u}_2(0)\| \le \epsilon$, which proves the continuity of l^{-1}. ■

Combining Propositions 2 and 3, we obtain the following theorem.

Theorem 1: The configuration space $\mathcal{C}$ of the rotating chain is homeomorphic to the 2-D surface $\mathcal{S}$ represented in Fig. 6.

B. Zero-Radius Loci and Low-Amplitude Regime

Before studying the stable subspaces, we need first to define the zero-radius loci and the low-amplitude regime in the configurations space.

Proposition 4 (and definition): Zero-radius loci are configurations whose corresponding attachment radii verify $r = 0$. Define $\bar{L}_i := L\omega_i^2/g$ where ω_i is the ith discrete angular speed (Appendix B). One has the following properties about the surface $\mathcal{S}$:

1) The ith zero-radius locus is an infinite curve that branches out from the $\bar{s}$-axis at $(\bar{L}_i, 0, 0)$, see Fig. 6;
2) The ith zero-radius locus separates configurations in rotation mode $i-1$ from those in rotation mode i.

Proof: 1) This property is implied by Kolodner's results, see the first paragraph of Section IV-C for more details.

2) Consider a rotation in mode $i-1$ and the corresponding curve $(\bar{s}, u_1(\bar{s}), u_1'(\bar{s}))_{\bar{s}\in[0,\bar{L}_1]}$. By definition, $u_1'(\bar{s})$ has $i-1$ zeros in the interval $[0, \bar{L}_1]$. Equivalently, we see that the 3-D curve $(\bar{s}, u_1(\bar{s}), u_1'(\bar{s}))$ crosses the first, second, ... $i-1$th zero-radius loci. Now, since the loci start infinitely near the $\bar{s}$-axis [cf. point 1) above] and extend to infinity, any curve continuously deformed from $(\bar{s}, u_1(\bar{s}), u_1'(\bar{s}))_{\bar{s}\in[0,\bar{L}_1]}$ will also cross the same loci.

Consider now another rotation, which is in mode i, and the corresponding curve $(\bar{s}, u_2(\bar{s}), u_2'(\bar{s}))_{\bar{s}\in[0,\bar{L}_2]}$. By Theorem 1, one can associate the two rotations with their endpoints $A := (\bar{L}_1, u_1(\bar{L}_1), u_1'(\bar{L}_1))$ and $B := (\bar{L}_2, u_2(\bar{L}_2), u_2'(\bar{L}_2))$ on the surface $\mathcal{S}$.

Assume by contradiction that there exists a continuous curve Γ that connects A and B without crossing the ith zero-radius locus. Consider the curve Γ' formed by the concatenation of $(\bar{s}, u_1(\bar{s}), u_1'(\bar{s}))_{\bar{s}\in[0,\bar{L}_1]}$ and Γ. By construction, Γ' is a continuous curve that connects the origin and B without ever crossing the ith zero-radius locus. This contradicts the first paragraph of point 2). We have thus established that the ith zero-radius locus separates configurations of rotation mode $i-1$ from those in rotation mode i. ■

Proposition 5 (and definition): The low-amplitude regime corresponds to configurations associated with infinitely small values of $u(\bar{s})$ and $u'(\bar{s})$, for all $\bar{s} \in (0, \bar{L})$.

1) The low-amplitude regime corresponds to points on the surface $\mathcal{S}$ that are infinitely close to the $\bar{s}$-axis (see Fig. 6).
2) Moreover, this regime corresponds to points on the parameter space $\mathcal{A}$ that have small values of a.

Proof: 1) It is clear that a low-amplitude rotation has $u(\bar{L})$ and $u'(\bar{L})$ infinitely small. Conversely, if $u(\bar{L})$ and $u'(\bar{L})$ are infinitely small, by the continuity of the mapping l^{-1} in the proof of Proposition 3, the initial condition a is also infinitely small. Finally, integrating from an infinitely small a will yield $u(\bar{s})$ and $u'(\bar{s})$ infinitely small for all $\bar{s} \in (0, \bar{L})$.

2) This follows from 1). ■

The low-amplitude rotations with zero attachment radius thus correspond to $(\bar{L}_i, \delta u, 0)$, $i \in \mathbb{N}$ for small values of $|\delta u|$. In the sequel, we shall refer to the ith small-amplitude rotation with zero radius as the point $(\bar{L}_i, 0, 0)$ instead of the more correct phase "$(\bar{L}_i, \delta u, 0)$ for small values of $|\delta u|$."

C. Stability Analysis

So far we have considered the space of all configurations of the rotating chain, that is, all solutions to the equation of motion (1). However, not all configurations are *stable*; in fact, experiments show that many are not. This section investigates the structure of the stable subspace—the subset of stable configurations— and discuss stable manipulation strategies.

To analyze the stability of configurations, we model the chain by a series of lumped masses, connected by stiff links, see Fig. 7.

Denote the position of the ith mass in the rotating frame $\{\mathcal{F}\}$ by $\mathbf{x}_i \in \mathbb{R}^3$. The attached end is fixed in $\{\mathcal{F}\}$ at $\mathbf{x}_N$. The state of the discretized chain is then given by a $6N$-dimensional vector consisting of the positions and velocities of the masses

$$\mathbf{y} := [\mathbf{x}_0, \dot{\mathbf{x}}_0, \ldots, \mathbf{x}_{N-1}, \dot{\mathbf{x}}_{N-1}]. \tag{23}$$

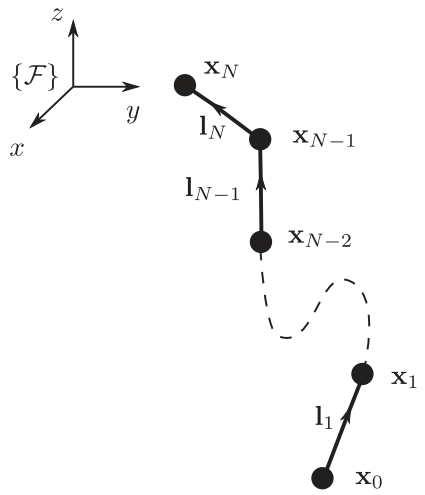


Fig. 7. Discretized chain model with N masses.

Applying Newton's laws to the masses (see details in Appendix D), one can obtain the dynamical equation

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}). \tag{24}$$

From Proposition 2, the configurations of the rotating chain can be represented by a pair $(a, \bar{L})$, which is associated with the position of the free end $\mathbf{x}_0$. Next, we discretize $(0, a_{\max}) \times (0, \bar{L}_{\max})$ into a 2-D grid. For each $(a, \bar{L})$ in the grid, we integrate, from the free end $\mathbf{x}_0$, the shape function of the discretized chain (23) at rotational equilibrium—in the same spirit as in Proposition 2. This discretized shape function corresponds to a state vector $\mathbf{y}^{\mathrm{eq}} := [\mathbf{x}_0^{\mathrm{eq}}, \mathbf{0}, \ldots, \mathbf{x}_{N-1}^{\mathrm{eq}}, \mathbf{0}]$. Finally, we assess the stability of $\mathbf{y}^{\mathrm{eq}}$ by looking at the Jacobian

$$\mathbf{J}(\mathbf{y}^{\mathrm{eq}}) := \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{y}}(\mathbf{y}^{\mathrm{eq}}).$$

Specifically, if the largest real part $\lambda_{\max} := \max_i \mathrm{Re}(\lambda_i)$ of the eigenvalues of $\mathbf{J}(\mathbf{y}^{\mathrm{eq}})$ is positive, then the system is unstable at $\mathbf{y}^{\mathrm{eq}}$; if it is negative, then the system is asymptotically stable at $\mathbf{y}^{\mathrm{eq}}$ [29, Th. 3.1].

Fig. 8(A) depicts the values of $\lambda_{\max}$ for $(a, \bar{L}) \in (0, 5) \times (0, 40)$. One can observe an interesting distribution of these values; in particular, the sharp transitions around the zero-radius loci (black lines). However, even though $\lambda_{\max}$ gets very close to zero on the left side of the zero-radius loci or in the low-amplitude regime, it is never negative, hinting that the system is at best marginally stable. While this could be expected from our model, which does not include any energy dissipation, it is contrary to the experimental observation of stable rotation states.

We need therefore to take into account aerodynamic forces in the chain dynamics, see details in Appendix D. Note that aerodynamic forces do not significantly affect the analysis of the previous sections, as their effect on the shape of the chain is negligible: for example, for a chain of length 0.76 m and parameters $(a, \bar{L}) = (2.0, 10.0)$, the changes in the equilibrium positions are less than 1 mm, which is 0.14% of the chain length.

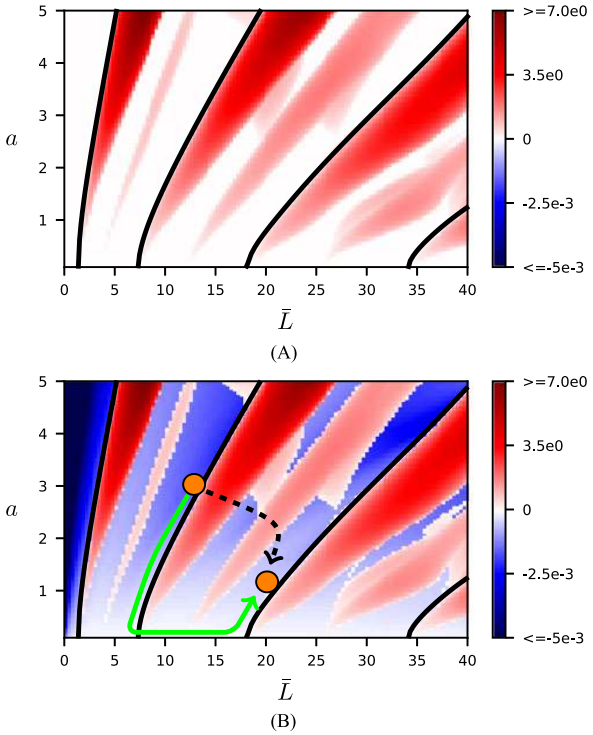


Fig. 8. (Best viewed in color) Maps of $\lambda_{\max}$, the largest real part of the eigenvalues of the linearized dynamics of two 10-link lumped-mass models at equilibrium: (A) model without aerodynamic forces and (B) model with aerodynamic forces. Positive values (red color) indicate unstable behaviors, while negative values (blue color) indicate asymptotically stable behaviors. Most configurations that are stable in the presence of aerodynamic forces cannot be concluded to be stable when there is no aerodynamic forces. Black lines: zero-radius loci—configurations whose attachment radii are zeros. Green arrow: A path in the chain's configuration space that contains only stable configurations. Black dashed arrow: A path that contains unstable configurations.

Fig. 8(B) depicts the values of $\lambda_{\max}$ for the system with aerodynamic forces. One can note that the overall distribution of $\lambda_{\max}$ is very similar to that of the system *without* aerodynamic forces [see Fig. 8(A)], but with the key difference that the regions in Fig. 8(A) with low but positive values now contain *negative* values of $\lambda_{\max}$ in Fig. 8(B) corresponds to asymptotically stable states.

One can make three more specific observations:

1) Configurations that are immediately on the right-hand sides of the zero-radius loci and with a relatively large are unstable (red color);
2) Configurations that are immediately on the left-hand sides of the zero-radius loci and with a relatively large are stable (blue color);
3) Configurations with a small (low-amplitude regime) are stable (light blue color).

Observation (1) hints that the upper portions of the zero-radius loci form "unstable barriers" in the configuration space. Therefore, it is *not* possible to stably transit between rotation

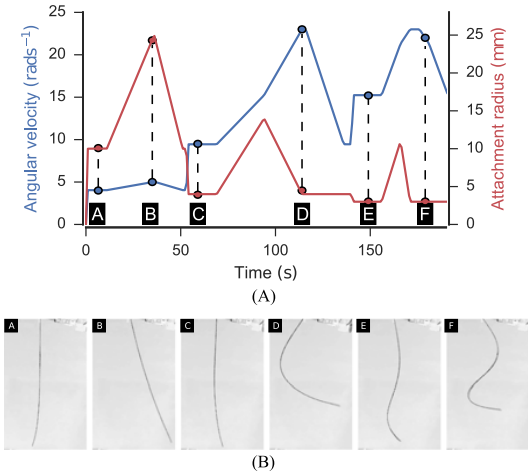


Fig. 9. A: Histories of the control inputs. Red: attachment radius r; blue: angular speed ω. A: low-amplitude rotation at critical speed ω_1. A $\rightarrow$ B: moving deep into rotation mode 0. B: stable rotation at mode 0. B $\rightarrow$ C: moving back to the low-amplitude regime with critical speed ω_1 and subsequently increasing the speed to ω_2 while staying in the low-amplitude regime. C: low-amplitude rotation at critical speed ω_2. C $\rightarrow$ D: moving deep into rotation mode 1. D: stable rotation at mode 1. E: low-amplitude rotation at critical speed ω_3. F: stable rotation at mode 2. Note that the attachment radius was not exactly zero in the low-amplitude regimes, but set to some small values. This was necessary to physically generate the desired rotation speeds. B: Snapshots of the chain at different time instants. The labels A–F refer to the same time instants as in the control inputs plot. A video of the experiment (including more types of transitions) is available at https://youtu.be/EnJdn3XdxEE.

modes $i-1$ and i (which requires crossing the ith zero-radius locus, see Proposition 4) while staying in the upper portion of the configuration space [dashed black arrow in Fig. 8(B)]. Observation (2) implies that transitions between configurations of the same mode can be stable. Observation (3) hints that a possible transition strategy might consist in 1) going down to the low-amplitude regime; 2) traversing the ith zero-radius locus while remaining in the low-amplitude regime; 3) going up toward the desired end configuration [see green arrow in Fig. 8(B)]. This strategy thus traverses only regions with negative $\lambda_{\max}$ and can be expected to be stable. The next section experimentally assesses this strategy.

VI. Manipulation of the Rotating Chain

A. Experiment

We now experimentally test the manipulation strategy proposed in the previous section. More precisely, to stably transit between two different rotation modes i and j, we propose to [see the green arrow in Fig. 8(B)]

1) Move from the rotation of mode i toward $(\bar{L}_{i+1}, 0, 0)$ while staying in the blue region of Fig. 8(B);
2) Move along the $\bar{L}$-axis toward $(\bar{L}_{j+1}, 0, 0)$;
3) Move from $(\bar{L}_{j+1}, 0, 0)$ toward the rotation of mode j while staying in the blue region of blue region of Fig. 8(B).

In practice, the histories of the control inputs (r and ω) to achieve the transitions in steps 1 and 3 can be found by simple linear interpolation, see, e.g., Fig. 9(A).

TABLE I
Critical Speeds for a Chain of Length 0.76 m

i	1	2	3
ω_i (rad s^{-1})	4.34	9.97	15.64

We performed the transitions

$$\text{Rest} \rightarrow \text{Mode } 0 \rightarrow \text{Mode } 1 \rightarrow \text{Mode } 2$$

on a metallic chain of length 0.76 m (note that the weight of the chain is not involved in the calculations). The upper end of the chain was attached to the end-effector of a 6-DOF industrial manipulator (Denso VS-060). The critical speeds, calculated using (36), are given in Table I.

A video of the experiment (including more types of transitions) is available at https://youtu.be/EnJdn3XdxEE. Fig. 9(A) shows the attachment radius and the angular speed as functions of time. Fig. 9(B) shows snapshots of the chain at different rotation modes. As can be observed in the video, the chain could transit between different rotation modes in a stable and controlled manner.

As the final note, we observed that any manipulation sequence that traverses highly unstable regions (red regions in Fig. 8) definitely leads to unsustainable rotations, as illustrated by the last section of the video.

B. Implications for Aerial Manipulation

For a circularly towing system, the ability to transit between rotation modes is desirable. Indeed, different modes have different functions. For instance, mode 0 rotations are most suitable to initiate a rotation sequence from a straight flying trajectory. On the other hand, rotations at higher modes such as modes 1 and 2 have more compact shapes, smaller tip radii, and higher tip velocities, and are therefore more suitable for performing the actual delivery or exploration.

It is furthermore desirable to switch modes in a quasi-static manner, as studied in this paper. Indeed, the *transient* dynamics of a heavily underactuated system such as the chain can be difficult to handle. The infinite dimensionality of the system, unavoidable modeling errors and aerodynamic effects make it challenging to design and reliably execute nonquasi-static mode-switching trajectories.

However, our result suggests that it is *not* possible to realize quasi-static mode transitions with fixed-wing aircraft. Indeed, since the turning radii of such aircraft are lower-bounded, the resulting rotations cannot enter the low-amplitude regime, which is necessary for quasi-static mode transition, as shown in the above development. Therefore, although nonquasi-static mode transitions are more challenging to plan and execute, they must be studied in future works.

VII. Conclusion

The study of the rotating chain has a long and rich history. Starting from the 1950's, a number of researchers have described

its behavior, and identified the existence of rotation modes. In this paper, we have investigated for the first time the *manipulation* problem, i.e., how to stably transit between different rotation modes. For that, we developed a framework for understanding the kinematics of the rotating chain with nonzero attachment radii and its configuration space. Based on this understanding, we proposed a manipulation strategy for transiting between different rotation modes in a stable and controlled manner. In turn, on the practical side, this result has some implications for aerial manipulation.

It can be shown (see Appendix A) that all the previous developments can be extended to the case of the chain with non-negligible tip mass. The key enabling notion here is that of *differential flatness* [13], with the flat output being the state of the free end. By differential flatness, given any trajectory of the free end, one can reversely compute the state trajectory and the control trajectory of the whole system. In fact, the property that we have "manually" discovered in this paper—the configuration space of a rotating chain is parameterized by the parameter space $\mathcal{A}$—might be related to the differential flatness of the rotating chain system. Indeed, each point $(a, \bar{L})$ corresponds to a circular motion of the free end, which in turn, by differential flatness, corresponds to the state and control trajectory of the whole chain, which in turn defines the configuration. This observation suggests two possible extensions as follows:

1) the motion of the free end can be more general (e.g., an ellipse), and can thereby lead to more practical applications, such as swinging to hit a target with the tip mass;
2) other differentially flat systems, whose flat output can be parameterized.

Another idea developed here, namely the visualization of the configuration space based on forward integration of the shape function, might find fruitful applications in the study of other flexible objects with "mode transition," such as elastic rods or concentric tubes subject to "snapping." Our future work will explore these possible extensions.

APPENDIX

A. Chain With Non-Negligible Tip Mass

Suppose that the free end of the chain carries a drogue of mass M. We show here that all the previous development can be extended to this more general problem.

We first proceed similarly to Section III and derive the dynamical equation of the rotating chain with tip mass. Writing the force equilibrium equation at the tip mass yields

$$F(0)z'(0) = Mg, \tag{25}$$

$$F(0)\rho'(0) = -M\rho(0)\omega^2. \tag{26}$$

Next, integrating (25) to obtain

$$F(s)z(s)' = g(\mu s + M) \tag{27}$$

where μ is again the linear density of the chain. This equation leads to

$$F(s) = g\frac{\mu s + M}{\sqrt{1-\rho'^2}}. \tag{28}$$

One arrives at the governing equation

$$\frac{\mathrm{d}}{\mathrm{d}s}\left(\rho'\frac{\mu s + M}{\sqrt{1-\rho'^2}}\right) + \rho\frac{\mu\omega^2}{g} = 0, \tag{29}$$

with boundary condition $\rho(L) = r$ where r is the attachment radius. One can now convert (29) to a dimensionless equation

$$\frac{\mathrm{d}^2 u}{\mathrm{d}\bar{s}^2} + \frac{u}{\sqrt{(\bar{s} + M\omega^2/\mu g)^2 + u^2}} = 0 \tag{30}$$

by the following changes of variable:

$$\begin{aligned} u &:= \rho'\frac{\mu s + M}{\sqrt{1-\rho'^2}}\frac{\omega^2}{\mu g}, \\ \bar{s} &:= \frac{s\omega^2}{g}. \end{aligned} \tag{31}$$

The boundary conditions are

$$u'(0) = a, \tag{32}$$

$$u(0) = a\frac{M\omega^2}{\mu g}, \tag{33}$$

$$u'(\bar{L}) = \bar{r} \tag{34}$$

where $a = -\rho(0)\omega^2/g$ and $\bar{r} = -r\omega^2/g$.

Equation (30) is a BVP that can be solved using the shooting method as described in Section IV-B. Moreover, we see that $(a, \bar{L})$ also parameterizes the solution space, which is the configuration space of the rotating chain with tip mass.

B. Low-Amplitude Regime

Here we recall the results obtained by Kolodner [1] for the low-amplitude regime. Low-amplitude rotations are defined by a zero attachment radius $r = 0$ and infinitely small values for the shape function ρ. Linearizing equation (9) about $\rho = 0$ yields

$$\rho w^2/g + \rho' + s\rho'' = 0, \tag{35}$$

with the boundary condition $\rho(L) = 0$.

By a change of variable $v := 2\sqrt{s\omega^2/g}$, one can rewrite the above equation as

$$\rho v + \rho_v + \rho_{vv} v = 0$$

which has solutions of the form

$$\begin{aligned} \rho(v) &= cJ_0(v), \quad \text{i.e.,} \\ \rho(s) &= cJ_0(2\omega\sqrt{s/g}) \end{aligned}$$

where J_0 is the zeroth-Bessel function. The boundary condition $\rho(L) = 0$ then implies that the angular speed can only take discrete values $(\omega_i)_{i\in\mathbb{N}}$ where

$$\omega_i = \frac{h_i}{2}\sqrt{g/L} \tag{36}$$

where h_i is the ith zero of the Bessel function J_0.

C. Useful Results From the Theory of Ordinary Differential Equations

Lemma 1 (Lipschitz): The ordinary differential equation (22) satisfies Lipschitz condition in some convex bounded domain $\mathcal{D}$ that contains $\mathcal{S}$.

Proof: Note first that $|u''(u,\bar{s})| < 1$ for all $u, \bar{s} \in \mathbb{R}$, which implies that $\mathcal{S}$ is bounded. Set now

$$\mathcal{D} := (0, \bar{L}_{\max}) \times (u_{\inf}, u_{\sup}) \times (u'_{\inf}, u'_{\sup})$$

where $u_{\inf}$, $u_{\sup}$, $u'_{\inf}$, $u'_{\sup}$ are bounds on $\mathcal{S}$. Clearly, $\mathcal{D}$ is bounded, convex, and contains $\mathcal{S}$. Next, all partial derivatives $\frac{\partial \mathbf{X}_i}{\partial x_j}$ are continuous in $\mathcal{D}$ (with continuation at $\bar{s} = 0$, see Remark in Section IV-A). This implies that $\mathbf{X}$ is Lipschitz in $\mathcal{D}$ [30]. ■

We now recall two standard theorems in the theory of Ordinary Differential Equations, see, e.g., [30].

Theorem 2 (Uniqueness): If the vector field $\mathbf{X}(\mathbf{u}, t)$ satisfies Lipschitz condition in a domain $\mathcal{D}$, then there is at most one solution $\mathbf{u}(t)$ of the differential equation

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{X}(\mathbf{u}, t)$$

that satisfies a given initial condition $\mathbf{u}(a) = \boldsymbol{c} \in \mathcal{D}$.

Theorem 3 (Continuity): Let $\mathbf{u}_1(t)$ and $\mathbf{u}_2(t)$ be any two solutions of the differential equation $\mathbf{X}(\mathbf{u}, t)$ in $T_1 \leq t \leq T_2$, where $\mathbf{X}(\mathbf{u}, t)$ is continuous and Lipschitz in some domain $\mathcal{D}$ that contains the region where $\mathbf{u}_1(t)$ and $\mathbf{u}_2(t)$ are defined. Then, there exists a constant M such that

$$\|\mathbf{u}_1(t) - \mathbf{u}_2(t)\| \leq e^{M|t-a|} \|\mathbf{u}(a) - \boldsymbol{y}(a)\|$$

for all $a, t \in [T_1, T_2]$.

D. Discretized Chain Model

Here we describe the procedure to obtain (24), which is the dynamical equation of the discretized chain model employed in Section V-C, see also Fig. 7.

The net force $\mathbf{F}_i$ acting on the ith mass is the sum of the following three components:

1) *fictitious forces*, which include the Coriolis force and centrifugal force associated with the rotating frame;
2) *constraint forces* generated by the ith and $i+1$th links;
3) *aerodynamic forces*, which include drag and lift.

Fictitious forces are computed using standard formulae, which can be found in any textbook on classical mechanics. To compute the constraint forces, we model the links as stiff linear springs whose stiffness approximates that of the chain used in the experiment of Section VI, which was $\simeq 8 \times 10^7\ N/m$. Constraint forces are then computed using Hooke's law.

Next, to compute aerodynamic forces, we follow the modelling choices of [18], i.e., the aerodynamic forces acting on the ith link, which is modelled as a cylinder, is placed entirely on the ith mass. Specifically, define the link length vector as $\mathbf{l}_i := \mathbf{x}_i - \mathbf{x}_{i-1}$ and denote by $\mathbf{v}_i$ the actual air speed of the ith mass. The angle of attack of the ith link is given by

$$\cos \xi_i = -\frac{\mathbf{l}_i \cdot \mathbf{v}_i}{\|\mathbf{l}_i\| \|\mathbf{v}_i\|}.$$

The drag and lift acting on the ith link are then given by

$$\mathbf{F}_i^D = 0.5 \rho_a C_D \|\mathbf{l}_i\| d \|\mathbf{v}_i\|^2 \mathbf{e}_D,$$
$$\mathbf{F}_i^L = 0.5 \rho_a C_L \|\mathbf{l}_i\| d \|\mathbf{v}_i\|^2 \mathbf{e}_L$$

where the directions and coefficient of drag and lift are

$$\mathbf{e}_D = -\frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}, \qquad \mathbf{e}_L = -\frac{(\mathbf{v}_i \times \mathbf{l}_i) \times \mathbf{v}_i}{\|(\mathbf{v}_i \times \mathbf{l}_i) \times \mathbf{v}_i\|},$$
$$C_D = C_f + C_n \sin^3(\xi_i), \qquad C_L = C_n \sin^2 \xi_i \cos \xi_i.$$

In the above equations, d denotes the diameter of the chain, C_f and C_n are, respectively, the skin-friction and crossflow drag coefficients, and ρ_a is the air density. As in [18], we use the following numerical values:

$$d = 1\,\mathrm{mm}, \qquad \rho_a = 1.225\ \mathrm{kg/m}^3,$$
$$C_f = 0.038, \qquad C_n = 1.17.$$

Summing the components, we obtain the ith net force $\mathbf{F}_i$, from which the acceleration of the ith mass can be found as

$$\ddot{\mathbf{x}}_i = \mathbf{F}_i / m_i$$

where m_i is the mass of the ith mass. Rearranging the terms, one obtains the dynamical equation (24)

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}).$$

REFERENCES

[1] I. I. Kolodner, "Heavy rotating string—A nonlinear eigenvalue problem," *Commun. Pure Appl. Math.*, vol. 8, no. 3, pp. 395–408, Aug. 1955.

[2] T. K. Caughey, "Whirling of a heavy chain," in *Proc. Third U. S. Nat. Congr. Appl. Mech.*, 1958, pp. 101–108.

[3] T. K. Caughey, " Large amplitude whirling of an elastic string—A nonlinear eigenvalue problem," *SIAM J. Appl. Math.*, vol. 18, no. 1, pp. 210–237, Jan. 1970.

[4] C.-H. Wu, "Whirling of a string at large angular speeds—A nonlinear eigenvalue problem with moving boundary layers," *SIAM J. Appl. Math.*, vol. 22, no. 1, pp. 1–13, Jan. 1972.

[5] C. A. Stuart, "Steadily rotating chains," in *Applications of Methods of Functional Analysis to Problems in Mechanics*. New York, NY, USA: Springer, 1976, pp. 490–499.

[6] J. J. Russell and W. J. Anderson, "Equilibrium and stability of a circularly towed cable subject to aerodynamic drag," *J. Aircraft*, vol. 14, no. 7, pp. 680–686, 1977.

[7] J. Toland, "On the stability of rotating heavy chains," *J. Differential Equations*, vol. 32, no. 1, pp. 15–31, Apr. 1979.

[8] D. J. Balkcom and M. T. Mason, "Robotic origami folding," *Int. J. Robot. Res.*, vol. 27, no. 5, pp. 613–627, 2008.

[9] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 249–267, 2012.

[10] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/unknotting manipulation of deformable linear objects," *Int. J. Robot. Res.*, vol. 25, no. 4, pp. 371–395, 2006.

[11] Y. Yamakawa, A. Namiki, and M. Ishikaw, "Dynamic high-speed knotting of a rope by a manipulator," *Int. J. Adv. Robot. Syst.*, p. 1, 2013.

[12] R. a. Skop and Y. I. Choo, "The configuration of a cable towed in a circular path." *J. Aircraft*, vol. 8, no. 11, pp. 856–862, 1971.

[13] R. M. Murray, "Trajectory generation for a towed cable system using differential flatness," in *Proc. IFAC World Congr.*, 1996, pp. 395–400.

[14] H. B. Gilbert, D. C. Rucker, and R. J. W. Iii, "Concentric tube robots: The state of the art and future directions," in *Proc. Int. Symp. Robot. Res.*, 2013, pp. 1–16.

[15] D. C. Rucker, B. A. Jones, and R. J. Webster, "A model for concentric tube continuum robots under applied wrenches," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 1047–1052.

[16] T. Bretl and Z. McCarthy, "Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 48–68, 2014.
[17] A. Borum and T. Bretl, "The free configuration space of a kirchhoff elastic rod is path-connected," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2958–2964.
[18] P. Williams and P. Trivailo, "Dynamics of circularly towed aerial cable systems, Part I: Optimal configurations and their stability," *J. Guidance, Control, Dyn.*, vol. 30, no. 3, pp. 753–765, 2007.
[19] M. Merz and T. A. Johansen, "Feasibility study of a circularly towed cable-body system for UAV applications," in *Proc. Int. Conf. Unmanned Aircraft Syst.*, 2016, pp. 1182–1191.
[20] H. Pham, J. H. Lim, and Q.-C. Pham, "Robotic 3D-Printing for Building and Construction," in *Proc. 2nd Int. Conf. Progress Additive Manufacturing*, Singapore, 2016, pp. 300–305.
[21] M. B. Colton, L. Sun, D. C. Carlson, and R. W. Beard, "Multi-vehicle dynamics and control for aerial recovery of micro air vehicles," *Int. J. Vehicle Auton. Syst.*, vol. 9, pp. 78–107, 2011.
[22] J. Nichols and L. Sun, "Autonomous aerial rendezvous of small unmanned aircraft systems using a towed cable system," *J. Guidance, Control, Dyn.*, vol. 37, no. 4, pp. 1–12, 2014. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.62220
[23] L. Sun, J. D. Hedengren, and R. W. Beard, "Optimal trajectory generation using model predictive control for aerially towed cable systems," *J. Guidance, Control, Dyn.*, vol. 37, no. 2, pp. 525–539, 2014.
[24] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load-a differentially-flat hybrid system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 4888–4895.
[25] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Auton. Robots*, vol. 30, no. 1, pp. 73–86, 2011.
[26] P. Williams, D. Sgarioto, and P. M. Trivailo, "Constrained path-planning for an aerial-towed cable system," *Aerospace Sci. Technol.*, vol. 12, no. 5, pp. 347–354, 2008.
[27] P. Williams and P. Trivailo, "Dynamics of circularly towed cable systems, Part 2: Transitiional flight and deployment control," *AIAA J. Guidance, Control, Dyn.*, vol. 30, no. 3, pp. 766–779, 2007.
[28] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, New York, NY, USA: Springer Science and Business Media, vol. 12, 2013.
[29] H. K. Khalil, *Noninear Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.
[30] J. Hu and W.-P. Li. (2005). Theory of ordinary differential equations. [Online]. Available: https://www.math.ust.hk/ mamu/courses/303/Notes.pdf

Hung Pham received the Bachelor's degree in mechanical engineering and is currently working toward the Ph.D. degree in robotics with Nanyang Technological University, Singapore. His research interests include in robotic motion planning and its applications to fields such as air traffic management and 3-D printing.

Quang-Cuong Pham was born in Hanoi, Vietnam, and brought up in Vietnam followed by France. He received the Graduate degree from École Normale Supèrieure, Paris, France, in 2007. He received the Ph.D. degree in neuroscience from Universitè Paris 6 and Collège de France, Paris, France, in 2009.

In 2010, he was a Visiting Researcher with University of Sào Paulo, Sào Paulo, Brazil. From 2011 to 2013, he was a Fellow with the Japan Society for the Promotion of Science, studying robotics at University of Tokyo. He joined Nanyang Technological University, Singapore, as an Assistant Professor in 2013.

ROBOTIC 3D-PRINTING FOR BUILDING AND CONSTRUCTION

PHAM TIEN HUNG
School of Mechanical and Aerospace Engineering,
Nanyang Technological University,
639798, Singapore

LIM JIAN HUI
School of Mechanical and Aerospace Engineering,
Nanyang Technological University,
639798, Singapore

QUANG-CUONG PHAM
School of Mechanical and Aerospace Engineering,
Nanyang Technological University,
639798, Singapore

ABSTRACT: In existing 3D printing processes, the size of the printed object cannot be larger than that of the built chamber or of the delivery system. This makes 3D printing a priori unsuitable for large-sized projects, such as in building and construction. In this paper, we present a framework based on mobile robotics to address that issue. More precisely, we propose to mount the delivery system on multiple robotic mobile platforms endowed with localization and precise placement capabilities. Doing so allows building objects of virtually any size, without compromising printing resolution or speed. We present the overall 3D printing pipeline, from robots placement optimization to localization and mapping, to path planning, to trajectory control and execution. We discuss the difficulties arising from both the materials delivery side and the robotic side. Finally, we give details about our hardware set-up and present some preliminary results.

KEYWORDS: robotics, 3D printing, building and construction

INTRODUCTION

Additive Manufacturing (AM) or 3D printing has been the focus of intense attention in recent years, with applications ranging from bio-engineering to automobile and aerospace manufacturing to food processing. In this paper, we explore the application of 3D printing in building and construction (B&C).

In traditional 3D printing processes, the size of the printed object cannot be larger than that of the built chamber or of the delivery system. This makes 3D printing a priori unsuitable for manufacturing large-sized objects, yet it is a requirement of B&C 3D printing. To address the issue, a mobile robotic framework is presented. The system is proposed to have the print mechanism mounted on multiple mobile robotic platforms. With locomotion, localization and precise placement capabilities, allows the printing of objects without dimensional constraints.

This paper is organized as follows. We first briefly review related works in 3D-printing-based B&C. Next, we present the overall pipeline of robotic 3D-printing for B&C. We then discuss the hardware and software architectures currently developed in our group to implement the pipeline. Finally, we conclude and sketch some directions for future work.

Proc. of the 2nd Intl. Conf. on Progress in Additive Manufacturing
Edited by Chee Kai Chua, Wai Yee Yeong, Ming Jen Tan, Erjia Liu and Shu Beng Tor

Published by Research Publishing, Singapore
ISSN: 2424-8967

RELATED WORK

On related works, we remark on their mechanical system without observations on their printing material. In literature, there are three main approach to 3D printing involving cement: Contour crafting (Khoshnevis 2004, Khoshnevis, Kwon et al. 2004, Zhang and Khoshnevis 2013), Concrete printing (Lim, Le et al. 2009) and D-shape (Le, Austin et al. 2012). Contour crafting and Concrete printing are primarily extrusion-based techniques, where a nozzle extrudes cement-based material at predefined location with a large gantry system. Separating the two techniques would their slight variation in nozzle designs. The other method, D-shape is a large scale 3D printing method in which sand and magnesium-based materials are bound together to create large stone-like objects. D-shape also employs a gantry system for delivery of the binder in a control environment.

Common in existing techniques is the use of the large scale gantry system for transport, placement and control of the material delivering print head. The apparent drawback of a gantry system is the necessary constraint on the dimension of the print, limiting it to be smaller than the printer, and also preventing its use in confined environments such as caverns. Secondly, on the gantry systems, with its belt/ chain driven print head positioning, it may be impossible to install additional independent print heads, placing a limit on the time require for a print job. Lastly, given the sheer size of the gantry system, and its need to be robustly secured, restricts its deplorability and makes installation time and labor intensive.

In our proposed mobile robotic system, the above constraints are addressed as follows (1) As the print head module is controlled by a robotic arm sitting on a mobile platform, the overall system remains compact and is able to print without restriction on the size of the print job. (2) The robotic solution allows multiple printers to work synchronously in the same work environment. (3) Its mobile platform allows it to be deplored with no installation necessary.

ROBOTIC PRINTING PIPELINE

The pipeline

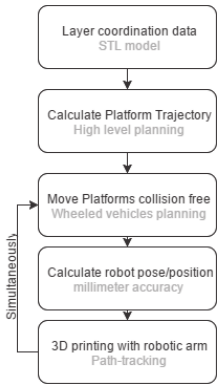


Figure 1: Process Pipeline

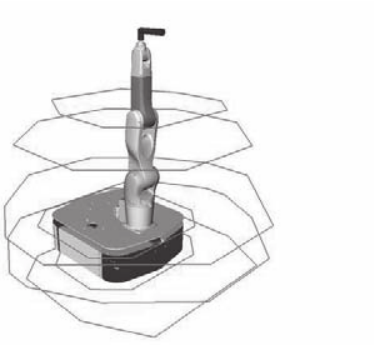


Figure 2: Printable area for the manipulator. Blue: printable areas, Red : non-printables because of collision or kinematic unreachability.

Figure 1 shows three steps of the pipeline. The first step is to generate printing paths from the 3D model of the desired object. This process depends heavily on the material deposition strategy employed. For example, in conventional extrusion 3D printing process, the object is sliced into thin, closely packed paths. In concrete printing, the path is not deposited from one single nozzle, but probably from a set of multiple nozzles with variable geometry and large area. In this case, a wall can be printed using a single print path.

The second step is to find some feasible sets of placements of robots. We define a placement as feasible if the robot in that placement can deposit material accurately into the desired printing paths. By formulating a geometrical optimization problem, we take advantage of the advanced numerical solvers to find the optimal value for positions and orientations of the robots.

The third step is to plan motion and control for robots to deposit material accurately onto the path in while avoiding collision. We devise our own algorithm to accomplish this task – task-precise coordinated motion planning. The idea is to decouple the problem into two simpler ones: 1) planning for individual robot 2) coordination and control for the whole system.

HARDWARE

In this section, we present the hardware components of our robotic setup. These include a robotic arm, a mobile platform and a 3D stereo camera.

Robotic arm and mobile platform

Our robotic system comprises of a six-axis position controlled robotic arm. The robotic arm is capable of 7 kg payload - which allows mounting of our complex in-house nozzle design. The reach of this robotic arm is 900 mm from its base, with a very high repeatability of 0.02 mm.

A mobile platform is used to house and transport the robotic arm in its workspace. This platform has an omni-directional Mecanum wheel mechanism, which allows it to move in any direction. Thus maximum flexibility is given to the robotic arm for material deposition material in the desired path.

Perception

The proposed robotic cement printer is equipped with a 3D stereo camera which employs a projected texture with active stereo vision method to create a point cloud model of the scene. From the point cloud model, depth information can be used for pose estimation and simultaneous localization and mapping (SLAM) problems.

The stereo camera is equipped with random point pattern projector which imposes artificial texture onto the image scene, solving the problem of homogeneous scene. The artificial texture induces interest points which facilitate the extraction of depth information based on epipolar constraints.

Modular print head

A modular extrusion print head is designed to be attached to the end effector of the robotic arm, with the idea of granting the proposed robotic printer the option of modular upgrades while the cost saving of retaining the core robotic framework.

SOFTWARE

In this section, we describe in details the software/algorithms in our framework.

Task-precise coordinated motion planning

The non-robotic audiences might find it beneficial to understand the concepts of path and trajectory in robotic. Both refer to how the joint values of a robot changes, the latter with respect to time and the former without time. Particularly, one can simply think of a path as a continuous line in the joint space of a robot. A trajectory is time-parameterized path, which is simply an indication of how fast it is being executed. One can easily see that it is possible to re-time a trajectory by keeping the path the same but changing how fast the robot moves along that path. This is the main idea behind our method.

In robotic 3D printing, multiple robots must coordinate to print/deposit material precisely at predefined velocity profile while avoiding collision. Using the terminology just introduced, we need to find the corresponding trajectories for all robot collision-free and precisely. We use a decoupled approach in which motion planning for robots is divided into two steps. Firstly, we plan trajectory for individual robot using a combination of randomized motion planning and closed-loop differential kinematics techniques, which will not be discussed here due to space constraint. This step will give us a trajectory for each robot, which might not be collision-free if execute together. Secondly, we re-coordinate the motions i.e. keeping the path of each trajectory the same but altering its time-parameterization so that they are collision-free while still, perform the tasks precisely.

Main idea behind the re-time procedure is to plan another trajectory in the coordination space. The coordination space can be thought of as a cube with perpendicular axes which are the time-intervals of each trajectory. Any point inside this cube corresponds to one instance at which the robots are at a particular combination of configurations. These configurations can be found from the coordinates of the point with respect to each axis, which is simply the time along the respected trajectory. By planning a trajectory in this space, starting at the initial time and ending at some end time, we obtain a full time-parametrization of all robots.

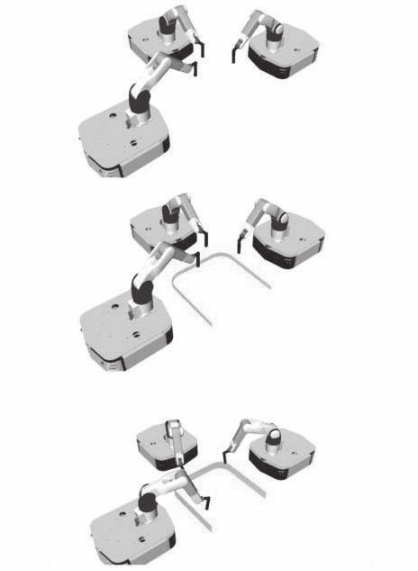


Figure 3: Multiple robots collaborate to print a single structure.

Placement optimization

Placement Optimization is the module that is responsible for the first step in our pipeline: calculating the optimal number and position of robots. We re-formulate the task into a geometrical optimization problem. In particular, the robot's reaching range is calculated and approximated as polygons (Figure 2). Next, we define n robots' orientation and position as variables $\boldsymbol{x}_i, 0 \leq i \leq n$ and formulate the cost function as follow

$$F(\boldsymbol{x}_1, \boldsymbol{x}_2, ...) = length\left(\bigcup_j l_j \setminus \bigcup_i \mathcal{R}(\boldsymbol{x}_i)\right) + collision\_cost(\boldsymbol{x}_1, \boldsymbol{x}_2, ...), \tag{6}$$

where $\mathcal{R}(\boldsymbol{x}_i)$ stands for the set of reachable space of the i-th robot, l_j stands for the j-th printing segment and function $collision\_cost(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots)$ is assigned some large value if collision happens and zero if there is no collision. We can formulate the following optimization

$$\begin{aligned} \underset{0 \leq i \leq n}{\text{minimize}} \quad & F(\boldsymbol{x}_i, ...) \\ \text{subject to} \quad & \boldsymbol{x}_i \in \mathrm{SE}(2) \end{aligned} \tag{7}$$

This optimization problem can be solved by general nonlinear solvers at a reasonable speed. While time might not be a problem, since the calculation of placement can be pre-processed using powerful servers. In our experiment, 3-placement problems can be solved in less than 2 minutes.

Robot pose estimation

Self-localization is a fundamental requirement for the autonomous robot printer, especially when multiple robots are synchronously printing in the same working environment. It is well understood that odometry alone is insufficient to solve the localization problem due to its unbounded uncertainty in pose estimation.
Through the stereo camera and ArUco fiducial markers set in the environment, the robot's pose estimates can be updated with observations of the world. In seeing the markers, relative pose between the camera and the seen object is obtained, and the robot can localize its position unambiguously.

The ArUco markers contain a 6x6 grid of which the four corners of the black border provide the necessary number of points for pose estimation with a calibrated camera. The black border also eases the detection of markers by providing strong contrast. The inner 5x5 region of the ArUco marker is used for identification of different marker IDs and error detection. Accuracy in pose estimation may be improved by arranging multiple markers in a board, which allows more points for the computation of the cameras extrinsic.

CONCLUSION

We have presented a framework for robotic 3D printing for construction. Our idea is to mount the delivery system on multiple coordinated mobile robots, which allows planning structures significantly larger than the delivery system itself, without compromising resolution and speed. Our future work consists in integrating the hardware and software architectures with the material delivery systems (nozzle, pump, printable cement, etc.) and demonstrating the capability of the overall framework on a typical construction task.

REFERENCES

Khoshnevis, B. (2004). "Automated construction by contour crafting—related robotics and information technologies." Automation in construction **13**(1): 5-19.

Khoshnevis, B., H. Kwon and S. Bukkapatnam (2004). Automated Construction using Contour Crafting. IIE Annual Conference. Proceedings, Houston, TX, USA.

Le, T. T., S. A. Austin, S. Lim, R. A. Buswell, R. Law, A. G. F. Gibb and T. Thorpe (2012). "Hardened properties of high-performance printing concrete." Cement and Concrete Research **42**(3): 558-566.

Lim, S., T. Le, J. Webster, R. Buswell, S. Austin, A. Gibb and T. Thorpe (2009). FABRICATING CONSTRUCTION COMPONENTS. USING LAYERED MANUFACTURING TECHNOLOGY. Global Innovation in Construction Conference 2009, Loughborough (United Kingdom), Loughborough University, Civil and Building Engineering.

Zhang, J. and B. Khoshnevis (2013). "Optimal machine operation planning for construction by Contour Crafting." Automation in Construction **29**: 50-67.

Automation in Construction 95 (2018) 98–106

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Large-scale 3D printing by a team of mobile robots

Xu Zhang, Mingyang Li, Jian Hui Lim, Yiwei Weng, Yi Wei Daniel Tay, Hung Pham, Quang-Cuong Pham*

Singapore Centre for 3D Printing, School of Mechanical & Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore

ARTICLE INFO

Keywords:
3D cementitious material printing
Additive manufacturing
Building and construction
Multi-robot
Large-scale 3D printing

ABSTRACT

Scalability is a problem common to most existing 3D printing processes, where the size of the design is strictly constrained by the chamber volume of the 3D printer. This issue is more pronounced in the building and construction industry, where it is impractical to have printers that are larger than actual buildings. One workaround consists in printing smaller pieces, which can then be assembled on-site. This workaround generates however additional design and process complexities, as well as creates potential weaknesses at the assembly interfaces. In this paper, we propose a 3D printing system that employs multiple mobile robots printing concurrently a large, single-piece, structure. We present our system in detail, and report simulation and experimental results. To our knowledge, this is the first physical demonstration of large-scale, concurrent, 3D printing of a concrete structure by multiple mobile robots.

1. Introduction

Compared to traditional construction techniques, 3D-printing (also known as Additive Manufacturing) carries the promise of faster, safer, more customizable, and less labour-intensive operations in multiple segments of the Building and Construction (B&C) industry [1]. Recent years have seen rapid developments in 3D-printing for B&C, from the formulation of printable materials [2–4], to the design of new printing systems [5–9], to commercialization [10,11].

A major hurdle to the widespread adoption of 3D-printing in B&C is the limitation on the sizes of the printed structures. As reviewed in detail in Section 2, most existing 3D-printing systems for B&C are based on a gantry, which can only print structures whose sizes are at most as large as that of the gantry itself. Some arm-based systems have been demonstrated, but the sizes of the printed structures in this case are limited by the reach of the robotic arm. One workaround consists in printing smaller pieces, which can then be assembled together. This workaround generates however additional design and process complexities, as well as creates potential weaknesses at the assembly interfaces.

To overcome this scalability issue, we propose in this paper a 3D-printing system based on a team of multiple mobile robots. Such a system can potentially print single-piece structures of arbitrary sizes, depending on the number of deployed robots. We demonstrate, for the first time to our knowledge, the actual printing of a single-piece concrete structure by two mobile robots operating concurrently (see Fig. 1 and video at https://youtu.be/p_jcG25tUoo). The size of structure is 1.86 m × 0.46 m × 0.13 m (length, width, height), which is larger than the reach of each robot arm taken separately (1.74 m), highlighting the need for multi-robot deployment. According to the classification method proposed in [12], where concrete 3D-printing techniques are classified based on object scale (x_o), extrusion scale (x_e), environment (e), assembly strategies (a) and support (s), our system of collaborative printing is categorized as $x_o^1 x_e^1 e^0 a^0 s^0$ with robotic complexity of r_6, which is higher than all state-of-the-art techniques as recorded in [12]. Note that concurrent printing is important to guarantee good bonding properties at the junctions: sequential printing would lead to fresh concrete adjoining hardened concrete at the junctions, weakening thereby the bonding strength [3,13].

Concurrent 3D printing by multiple mobile robots is difficult for several reasons. First, the robot motions must be carefully planned and coordinated to optimize material delivery while avoiding mutual collisions. Second, robot localization must be highly precise to ensure that the pieces printed by different robots are perfectly aligned. Finally, the mixing and pumping systems of the robots must be coordinated to deliver materials in a synchronized manner.

The remainder of the article is organized as follows. In Section 2, we review existing 3D-printing systems for B&C. In Section 3, we present in detail our system based on a team of mobile robots. In Section 4, we report the results of the multi-robot printing experiment. In Section 5, we discuss the advantages and limitations of the proposed system. Finally, in Section 5.1, we conclude and sketch some directions for future

* Corresponding author.
E-mail address: cuong@ntu.edu.sg (Q.-C. Pham).

https://doi.org/10.1016/j.autcon.2018.08.004
Received 29 April 2018; Received in revised form 20 July 2018; Accepted 10 August 2018

Fig. 1. Concurrent printing of a large, single-piece, concrete structure by two mobile robot printers. See the full video of the experiment at https://youtu.be/p_jcG25tUoo.

work.

2. Related works

2.1. Material development of 3D cementitious material printing

In recent years, various 3D concrete printing materials have been developed, categorized primarily into 3D printable plain concrete [2,3,14], 3D printable geopolymer [15], 3D printable fibre reinforcement concrete [16,17], 3D printable rapid hardening materials [18,19] and 3D printable earth-based materials [20]. These materials share a common emphasis on their rheological performances, which directly impacts the buildability and printability of 3D concrete printers, quantified by measurements such as printed height and pumping pressure respectively.

In the literature, several material models have been developed to understand material behaviour. Perrot et al. first established a model correlating yield stress and geometric factor to buildability [21], and his study was extended by Weng et al. towards more realistic application [22]. Weng's built-up model predicts the buildability of hollow cylinder using material static yield stress and geometric factor of the printing design. Wolfs et al. have shown that the elastic properties evolution is also critical in order to avoid structure collapse by buckling [23]. In another rheology study, Chhabra et al. proposed a model relating pumping pressure to rheological performance [24]. His model indicates that pumping pressure is governed by material plastic viscosity. Concrete is thixotropic [25] due to its continuous hydration, and this mean that its viscosity becomes less viscous when undergoing shear stress due to pumping. From these earlier works done, it is clear that the rheological property and elastic properties evolution of concrete are essential factors affecting concrete printing in terms of buildability and pumpability.

2.2. 3D concrete printing systems

Concrete printing systems can be divided based on their system mechanism, which is primarily gantry systems and robotic arm systems.

2.2.1. Gantry-based systems

The gantry system is widely adopted and uses a gantry to position the print nozzle in XYZ Cartesian coordinates. The build envelope of the gantry system is determined by enclosed volume of the gantry. A number of notable gantry systems include Contour Crafting [5,26–28], Concrete Printing [2,29,30] and D-shape [6].

These three techniques differ in their printing technique, with Contour Crafting and Concrete printing using an extrusion-based method similar to the Fused Deposition Method in additive manufacturing, and D-shape uses a binder jetting technique to selectively deposit binders on a powder bed made up of magnesium-based materials and sand. Another difference between Contour Crafting and Concrete printing is Concrete Printing's use of printing supports, which allows Concrete Printing to print full 3D topology as compared to Contour Crafting's vertical extrusion of a planar shape.

2.2.2. Arm-based systems

Robotic arm systems are relatively new compared to the gantry system counterparts. They provide additional roll, pitch and yaw controls to the end effector (print nozzle), allowing the print nozzle to perform more articulate print designs, such as printing with the tangential continuity method [7]. The tangential continuity method allows a smoother transition between print layers by maintaining a continuous rate of curvature change, giving a more aesthetically pleasing look. Another robotic arm system by Keating et al. [8] in Digital Construction Platform (DCP), they mounted the robotic arm on a track driven mobile platform for on-site fabrication of printed structures. DCP's system is also self-sufficient by recharging its electrical drive system with solar panels. One other mounted robotic arm system is Cybe RC 3Dp [31] which has a 6-axis robotic arm mounted on caterpillar tracks and is used in 3D printing the R&Drone Laboratory in Dubai [11].

2.2.3. Minibuilders

Minibuilders [9] presents an alternative approach for 3D concrete printing. They use three small mobile robots in the system. The first robot is equipped with a sensor that follows an initial marked path and builds the concrete foundation. The second robot is placed on the foundation and gripped the foundation with rollers before printing additional layers of concrete, and building up the structure. The last robot uses suction cups and pressurized air to print vertically up the printed structure and reinforced the printed structure which had only horizontal layers.

2.2.4. Summary

The biggest limitation in literature of printing system is their lack of scalability. Gantry-based system and stationary robotic arm system requires a massive external framework to support the single print nozzle in building the structure. While mobile robotic arm system helps extend the printing range, a single print nozzle still hoards the entire print space, limiting the efficiency of the printer. Although a multiple agent system is introduced by Minibuilders, their robots requires a harden structure for climbing and therefore has limited application as it involve waiting for the printed concrete to grant sufficient strength before deployment.

This gap in scalability motivates our project of a multi-robot printing system. Our system utilizes multiple mobile robot printers in a multi-agent setting to print their individual portion of a large print structure. They are capable of localization, collision avoidance and efficient coordinated printing through optimal robot placement. Our system demonstrate scalability by allowing users to introduces as many robots as needed in a shared environment for task completion in a fast and efficient manner.

2.3. Robotics background

Some robotics knowledge of the algorithms used in our proposed system for robot localization and planning is introduced in this section.

2.3.1. Simultaneous localization and mapping (SLAM)

Self-localization is critical for mobile robots to allow it to navigate its environment. The common strategy for localization in an unstructured environment involves using Simultaneous localization and mapping (SLAM) to construct and or update the map of the environment while keeping track of the robot in the environment [32–34]. A recent literature survey on state-of-the-art methodologies for

implementing solutions to the SLAM problem can be found in [35].

2.3.2. Motion planning

A fundamental motion planning task is to plan collision-free motions for a robot to move from a start to a goal position among a collection of obstacles [36]. The problem is often solved in configuration space, which is the set of all possible configurations for the robot.

Low-dimensional problems can be generally solved using grid-based algorithms that overlay a grid on top of configuration space and map each configuration to a grid point, or geometric algorithms where shape and connectivity of free space are computed. However, sampling-based algorithms are commonly used for high-dimensional problems due to their higher computational efficiency, where configuration space is represented with a road map of sampled configurations.

When it comes to path planning for multiple robots, there are basically two approaches: coupled and decoupled. A coupled planner treats the robots as a single combined robot and computes a path in a combined configuration space.

3. Large-scale 3D printing by multiple mobile robots

3.1. System setup

Each mobile robot printer in our setup consists of a holonomic mobile platform, a 6-axis robotic arm, a stereo camera and a pump as shown in Fig. 2. The robotic arm is mounted on the holonomic mobile platform and is equipped with a print nozzle. The robotic arm has a reach of 0.9 m with a repeatability of 0.02 mm. The holonomic mobile platform is equipped with sensors for localization and odometry, which includes wheel encoders, inertial measurement unit (IMU) and a 2D laser scanner. Localization is performed using on-board sensors, the stereo camera and some ArUco markers [37] placed on the platform. The markers have a black border for fast tracking and an inner binary matrix which determines its identifier, preventing feature mismatch. The pump system is responsible for delivering cementitious material to the print nozzle for selectively concrete deposition to build up the final print structure.

3.2. Pipeline

The printing pipeline is illustrated in Fig. 3.

Firstly, for a given 3D CAD model, it was sliced into thin, closely packed paths in a layer wise fashion. The coordinates and normal of the path were extracted as way points for the robotic arm. Using the geometry information of each layer, the multi-robot placement situation was formulated as an optimization problem, which was solved for finding feasible and reasonable pose for each corresponding robotic manipulator. A feasible solution is defined if a robot is its placement can deposit material accurately onto the desired print path without self-collision or exceeding its joint limit, while a reasonable solution is defined if the robot is able to avoid potential failure points if there is a better solution. The algorithm for multi-robot placement optimization will be illustrated in Section 3.4.

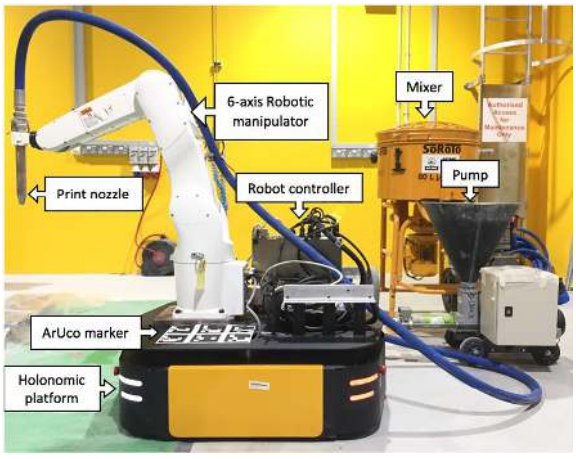


Fig. 2. System setup for one robot printer (stereo camera is placed out of the scene).

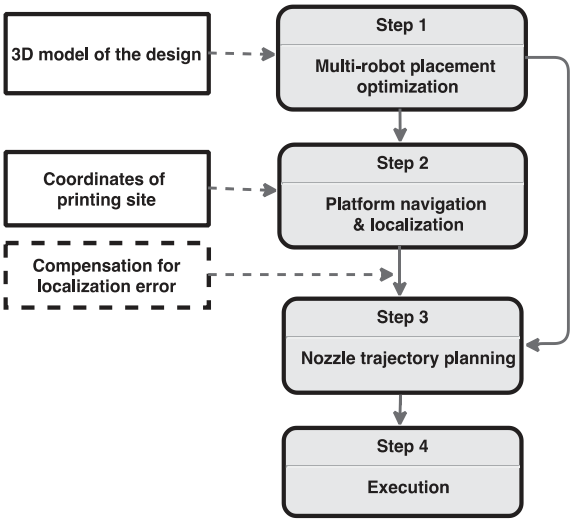


Fig. 3. Printing pipeline.

Secondly, with the construction environment and robot placement, the mobile robots would navigate to their respective target position accurately and without collision. In this phase, an environment map is generated with the on-board sensors and feasible paths are planned for navigation. When the mobile robots are in proximity to their target position, they are guided to their final position, using the stereo cameras. This methodology will be explained in Section 3.5.

Subsequently, the nozzle trajectory is planned so that it can extrude cementitious material accurately onto the required path in a time-coordinated manner in order to avoid collision with the environment, the printed structure and the other robot printers. This trajectory planning phase is done in parallel while the robot printer is navigating its environment. The nozzle trajectory planner takes into account the individual sub-workspace of each robot and the coordinates and normal of the way points generated in the first module, while correcting for minor error in the mobile platform localization. The algorithm for the multi-robot motion will be discussed in Section 3.6.

Finally, with the robot printers in their respective position, the printing process will begin.

3.3. Materials preparation and extrusion methods

The cementitious materials were designed to meet rheological requirements (yield stress and viscosity) for printing, and this was reflected by pumpability during the material delivery phase and buildability during the extrusion phase. Two separate materials were chosen to demonstrate the adaptability of the system towards multi-material printing. One material consists of ordinary concrete with its mix design shown in Table 1, and the other material is a fibre reinforced concrete mix design [22] and shown in Table 2.

Two SoRoTo Forced Action Mixer 80 L were used in mixing. As mixing factors, such as mixing time, speed and temperature, have been known to affect cementitious material rheological properties, mixing

Table 1
Mixture proportion for robot printer 1 (kg/m^3).

Materials	Cement (OPC, ASTM Type 1)	Fly ash (Class F)	Silica fume (Undensified, Elkem)	River Sand	Water
Proportion	613.7	287.7	57.5	767.2	402.8

Table 2
Mixture proportion for robot printer 2 (kg/m^3).

Materials	OPC	Sand	Water	Fly ash	Silica fume	Superplasticizer
Proportion	575.7	590	330.3	575.7	863.4	1.4

Note: All ingredients contents are expressed as weight proportion of cement content.

procedures were strictly standardised and adhered to, so as to ensure the consistency of rheology performance. Firstly, the powder of all solid ingredients was dry mixed for 1 min at stir speed; water was then added and mixed for another minute at stir speed. Superplasticizer was added, and the mixing speed was increased to speed I for 1 min, and speed II for 1 min. Finally, the fibre is introduced, and the mixing process continued for 2 min at speed II. On completion of the mixing procedures, the cementitious material was loaded into the pumps' hoppers.

During printing, a rotor/stator pump was used to deliver the cementitious material through a 5 m long 25.4 mm diameter hose pipe at 650 rpm. The cementitious material was extruded through a 10 mm tapered diameter print nozzle to build up the designed structure in a layer wise manner. The choice of a round nozzle also eases the orientation constraint on the nozzle allowing a wider pose solution to be found, which translates to a bigger printing volume by the robot printer.

3.4. Robot placement optimization

In literature, there are multiple studies on collision avoidance for multiple robots working in a shared workspace [38–40] or optimal robot placement given a prescribed task [41,42] but not the combination of two situations which is required in our multi-robot printing system. The multiple robotic printers have to cover the complete work piece in a collision-free environment yet no overlap of their individual workspace is allowed. We previously formulated this situation as an optimization problem that tries to minimize the length of work piece that is not able to be covered.

However, just satisfying workspace coverage might result in some robot configurations that would cause potential planning problems. In this work, we further improve the algorithm by introducing a score scheme for robot configuration considering its kinematic reachability.

The kinematic reachability of a point in the robot workspace is given by the number of possible robot configurations which realize this point. A kinematic reachability database for one robot printer was generated prior to printing (See Fig. 4).

Kinematic reachability was then normalized to the interval from 0 to 1 and was incorporated to an optimization program to find the optimal robot placement. In general, let N denote the number of robot printers, the optimization objective function is given by

$$\underset{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N \in SE(2)}{\operatorname{argmin}} \; -\sum_{i=1}^{N} \boldsymbol{S}(\boldsymbol{x}_i) + \boldsymbol{C}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N) \tag{1}$$

where $\boldsymbol{x}_i$ is the pose of the i-th robot printer, $\boldsymbol{S}(\boldsymbol{x}_i)$ evaluates the total reachability of the i-th robot printer regarding the target object, $\boldsymbol{C}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is the cost of collision between the robots, which is assigned to 0 if the system is collision-free and a large value if collisions exist.

3.5. Robot localization

Among the myriad of SLAM algorithms, we have adopted GMapping [43] for map construction because of its high efficiency and accuracy. Fig. 5(a) shows a map of our lab constructed with the sensors on board of the platform using GMapping. The constructed map was then used in adaptive Monte Carlo localization (AMCL) [44] for robot localization.

In Fig. 5(b) the rainbow colour thin lines were from the sensor data, while the blue regions are places considered occupied on the occupancy grid map. It showed the situation that the mobile robot succeeded in localizing itself in the map using AMCL algorithm, with the sensor observations fitting the given map well.

With the SLAM and AMCL algorithms, mobile platforms are able to localize themselves in a rough-scale and navigate in the environment at a low cost. However, this is far from the precision required for them to print concurrently on one work piece. Hence, we switched to vision control when the platform is near to the target printing site for higher accuracy. ArUco markers were attached on the mobile platforms and stereo cameras were installed around the construction site, providing easy and unique features for the camera to identify and avoid feature mislabels when using nature features in the environment. Upon observation of the markers, relative pose between the camera and the detected marker is calculated, and the robot can localize its position unambiguously.

3.6. Multi-robot motion planning

Motion planning is required for robot navigation and nozzle trajectory planning for multiple robot printers. In this work, we use decoupled planning approach since it is generally faster. Before talking about planning algorithms, it would be helpful to know that in the syntax of robotics, trajectory is a time-parametrized path, which indicates how fast a path is executed. We first compute a path for each robot independently, then use a coordination diagram to plan collision-free trajectories for each robot along its path. The second step can be thought of as a re-timing process, which is analogical to finding a path from (0, 0, …, 0) to (l_1, l_2, …, l_n) of a virtual cube with perpendicular axes representing trajectories of the robots. Each point inside the cube corresponds to one combination of robot configurations at particular time instance.

The above stated trajectory planning is a naive process which assumes perfect execution of the trajectory. However, for platform navigation the uncertainty can be significant due to the slippery of the wheels and imperfect condition of the floor, especially for construction environment. Therefore, to avoid collision of the mobile platforms and to navigate to the goal positions accurately, the platforms broadcast their locations at certain frequency for collision checking, and the trajectory is re-planned upon update of new data.

3.7. Actual printing

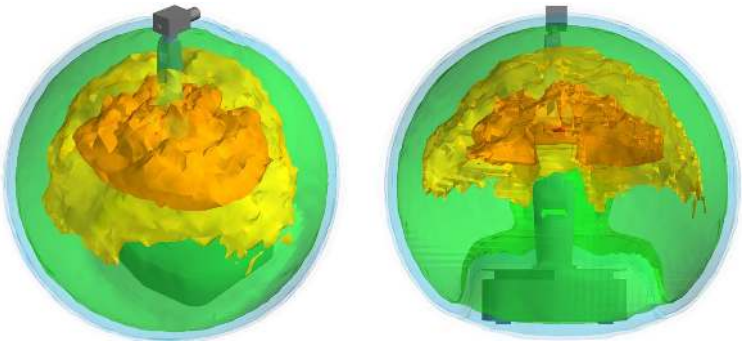
We used Ubuntu operating system for system integration [45], ROS for device control, message-passing between processes and package management [46], and OpenRAVE for planning and collision checking.

Prior to printing, a kinematic reachability database of the robot printers was generated. Subsequently, a map of the printing site was constructed with GMapping by moving a robot printer around the printing site. Stereo cameras were also set up around the site.

By slicing a 3D model of the target object, the waypoints, each consists of XYZ coordinates and a normal vector, were generated. This can be carried out using common slicing programs for 3D printing.

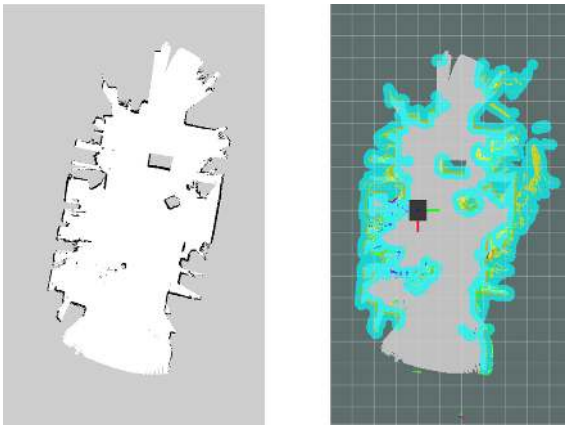
Next, taking as inputs the target printing pattern, the robot printers' models and their kinematic reachability databases, the algorithm in Section 3.4 was used to compute the desired placement of the robot printers.

Subsequent to robot placement optimization, trajectories for the



(a) Isometric view (b) View along the x

Fig. 4. Kinematic Reachability of a robot printer. The colours represent different levels of kinematic reachability: green and orange denoting the highest and lowest levels respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Map of the printing site constructed by GMapping. (b) Robot printer localization in the map achieved by AMCL

Fig. 5. Map construction and robot printer localization.

print nozzles are planned in OpenRAVE so that they can deliver material accurately on the desired paths without causing collisions with printed parts and other robot printers. Concurrently, the mobile printers plan their trajectories from home station to target printing positions using the map of the printing site constructed in earlier stage.

Upon completion of the planning, mobile printers navigate towards their target printing locations, using AMCL algorithm to track the mobile printers. In an effort to increase the robot localization accuracy, when the printers are within a pre-defined proximity to their target position, they will switch to vision-feedback control using the stereo cameras to track the ArUco markers placed on the mobile platforms. On reaching their target locations, the printers start to print on the work piece concurrently following the planned paths. The planned paths may be re-planned at this stage to compensate for any minor error in platform localization as discussed in Section 3.2.

Upon completion of printing, the robot printers will return to their home station, using the same navigation strategies when moving to the printing location. In the entire process, the robot printers will broadcast their status for collision checking, and also to synchronize the printing with the pumping system.

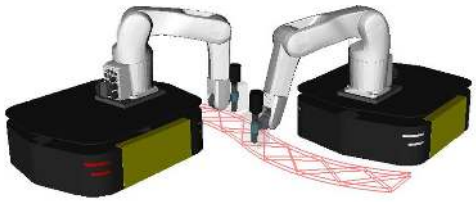
4. Results

Results of both simulation and demonstration on actual robotic system are presented in this section.

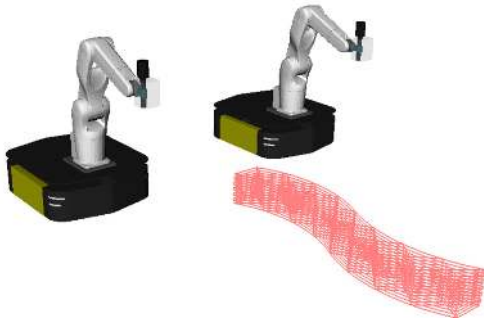
4.1. Simulation

In the simulation as shown in Fig. 6, two robot printers were required to build a large-scale structure whose size is beyond the printing volume of one single robot printer.

It took 24 s to plan for mobile platform navigation from home station to target printing locations, and 25 s from printing location back to home station. Planning of trajectories for the print nozzles to print 12 layers of the target structure took 11 s.



(a) Concurrent printing of one large-scale structure by two mobile robot printers.



(b) Robot printers back to home station upon completion of printing.

Fig. 6. Simulation of multi-robot printing process.

4.2. Execution

The simulation in Section 4.1 has been implemented on actual robotic system with the proposed printing pipeline. Fig. 7 presents an ordered sequence of intermediate steps during the printing process.[1]

Given target printing location and printing pattern, placement of robot printers and their corresponding printing paths were simulated. The two robot printers navigated from home station to their respective printing locations. Subsequently they started printing concurrently on the large structure in a coordinated fashion without collision. Upon completion of the print, the robot printers return to the home station.

4.3. Printed specimen

The printed structure was 1.86 m × 0.46 m × 0.13 m (length, width, height). The specimen was covered by a plastic sheet to simulate field conditions for ambient curing. After 10 days of curing, we could turn it over, as shown in Fig. 8.

The mechanical properties of the 3D printed concrete with each of the two materials can be found in [4,22]. Regarding the junctions, we did not perform tests (since this would involve cutting the specimen), but noted that the bonding strength was high enough to easily support the specimen's own weight when it was placed on its side (Fig. 8(b)). Informal visual and stress inspections suggested that the bonding was strong. Extensive and quantitative tests will be performed as part of our future work.

5. Discussion

The printing demonstration in Section 4.2 serves as a proof of concept for the multiple mobile robots printing system for printing concrete structure in an efficient and scalable manner. The printed design was chosen to incorporate complex curves and internal void that would be challenging to build using traditional construction methods. The size of the printed design was also chosen to extend beyond the workspace of a single robot printer, requiring the concurrent printing by multiple robots. With the decoupling of printer system from print material, it allows flexibility for users in selection of materials for different applications such as printing a load bearing structural wall to wall fillers. Our demonstration has intentionally used two different materials in showing its non-reliance on material type, as mentioned in Section 3.3.

As our proposed system has the attribute of arm-based system, mobile platform and swarm printing, its major advantages over gantry-based system, stationary printer and single printer are discussed here.

5.1. Advantages over gantry-based system

Robotic arm printers benefit from the greater flexibility of having full 6 degrees of freedom as compared to 3 or 4 degrees of freedom in the conventional gantry system.

This extra articulation allows the robotic arm printer to print complex curved parts using the tangential continuity method [7,47]. This results in a smoother transition between print layers by maintaining a continuous rate of curvature change, hence giving a structurally more efficient mechanical stability by reducing shear loads between printed layers.

5.2. Advantages over stationary printer

The mobile platform of our robot printer extends the reachability of the printer, by allowing the printer to print beyond the fixed environment it will otherwise be confined to when stationary. On-site robotic 3D printing is expected to alleviate transportation cost, as printing large structures on site helps to relieve pressure of transporting oversized parts that would not fit on a conventional truck.

5.3. Advantages over single printer

Swarm printing affords greater scalability and efficiency as compared to a single printer. It has been shown in our demonstration that swarm printer can build structure whose size (1.86 m × 0.46 m × 0.13 m) is larger than the reach of a single printer (1.74 m). When the same structure is to be printed with a single printer, it took twice the required duration of two printers.

However, there are also challenges for the system to work robustly for application in B&C industry. A major technical difficulty would be accurate localization of the robot printer under different terrain conditions. When the terrain is bumpy and dusty, which is common to a construction site, the mobility scheme for the mobile printer plays a vital role in the printing system. As Mecanum wheel designs for holonomic mobiles can perform poorly on rough terrains [48], it would be necessary to investigate alternative wheel schemes to enhance the robustness of the printing, such as differential tank drive.

6. Conclusion and future works

6.1. Conclusion

In this paper, we have demonstrated a multi-robot printing system that is capable of on-site printing large structures in a safe, efficient and scalable manner. The system configuration contains several modules: planning of robot placement to optimize workspace, mobile robot navigation and localization to reach target printing location, and planning of manipulator trajectory to deposit material accurately on desired path.

We presented our results of both simulation and demonstration on actual robotic system: two mobile robot printers work concurrently on one structure whose size is twice the printing volume of one single printer.

Comparing to existing 3D cementitious material printing techniques, experimental results suggest that the proposed system is superior in its greater practical scalability and higher time efficiency due to the employment of multiple robot printers, as well as better on-site printing capability credited with mobility of the system. These advantages make the proposed system promising to be scaled up for on-site large-scale printing applications in B&C industry. Both technical and economic benefits can be evaluated by the Construction Robotic Equipment Management System (CREMS) [49–52]. Furthermore, using a fleet of mobile robots for construction could have an extreme potential in other non-conventional aspects. One such application is to allow automated construction in hard-to-reach, remote areas, such as underground caves, the Moon or Mars [53], to which it is inconvenient or even impossible to bring other kinds of machine required for existing cementitious material printing methods.

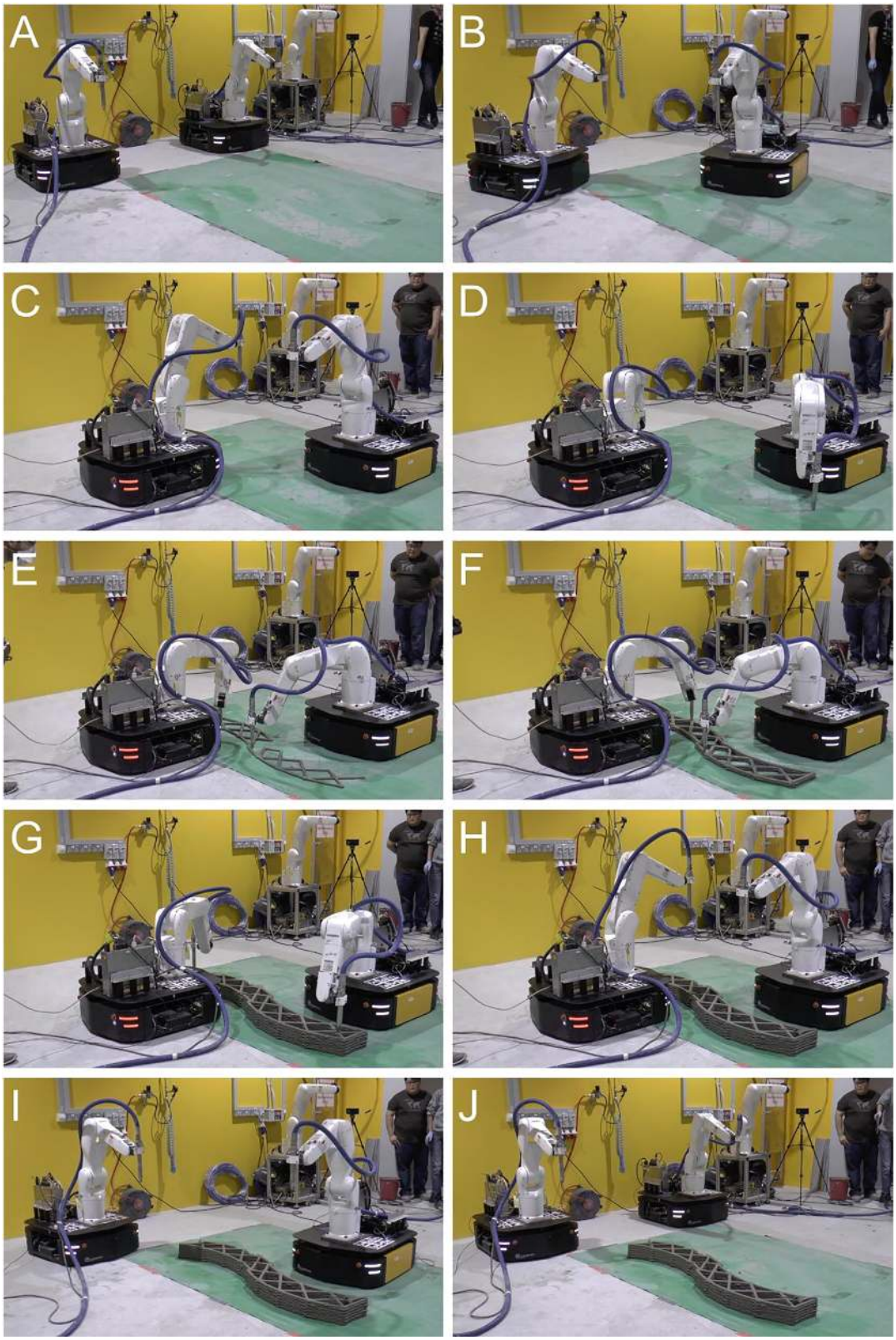
A technical concern of the proposed system is the complexity in coordinating such a fleet of mobile robots. However, this paper demonstrates that this difficulty can be significantly mitigated by utilizing standard robotic techniques.

6.2. Future works

With our successful proof-of-concept demonstration of the proposed multi-robot printing system, our primary future plan is to integrate the overall system to boost automation to a greater extent.

Moving forward from current printing-on-arrival scheme, where robot printers navigate towards desired printing locations and start to print at these stationary locations, our future research direction will

[1] Timings taken from the Supplement video material: 10 s to move to printing position, 470 s for printing (7 min 50 s), 10 s to move back to home station.



(caption on next page)

Fig. 7. Snapshots taken during the actual printing process. See the full video of the experiment at https://youtu.be/p_jcG25tUoo. A. Robot printers at home station; B. Navigating towards target locations; C. Reaching target locations; D. Starting printing; E. Printing in progress; F. Building up layers; G. Finishing printing; H. Robotic manipulators back to standby pose; I. Navigating back; J. Reaching home station.

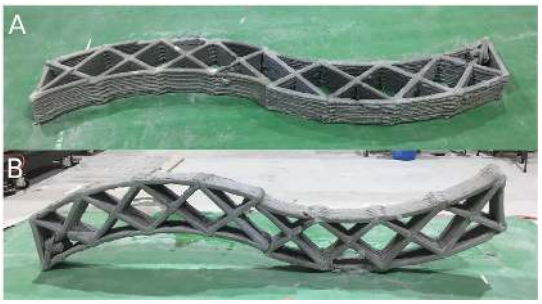


Fig. 8. Printed specimen after 10 days of curing.

focus on study of printing-while-moving scheme to fully utilize the advantage of mobile robot printer and further extend the printing scale. On account of the integration of printing-on-arrival and printing-while-moving, the study will also involve investigation of trade-off between workspace size covered by each single mobile robot printer and inter-layer strength of 3D printed concrete.

Apart from exploring alternative wheel schemes, future work would include integration of additional sensors that aids in the online determination of nozzle offset distance to compensate for changes in ground level while printing.

In addition, bonding strategies at the interface of two separate concrete prints would be carried out to identify the optimal method of joining, which could be face-to-face joint or finger joints. Such methods would seek to reduce the degree of material overlapping that could adversely affect part appearance while maximising interface area for stronger bond.

Supplementary data to this article can be found online at https://doi.org/10.1016/j.autcon.2018.08.004.

Acknowledgement

This research is supported by the National Research Foundation, Prime Ministrer's Office, Singapore under its Medium-Sized Centre funding scheme, Singapore Centre for 3D Printing, and Sembcorp Design & Construction Pte Ltd. The authors would like to thank Y. Qian, B. Panda, W. Lao, Z. Liu, A. Ting, and A. Annapareddy for their helps with the experiment, Prof. S. Qian and Prof. M. Tan for helpful discussions.

References

[1] Y.W.D. Tay, B. Panda, S.C. Paul, N.A. Noor Mohamed, M.J. Tan, K.F. Leong, 3D printing trends in building and construction industry: a review, Virtual Phys. Prototyp. 12 (3) (2017) 261–276, https://doi.org/10.1080/17452759.2017.1326724.

[2] T.T. Le, S.A. Austin, S. Lim, R.A. Buswell, R. Law, A.G. Gibb, T. Thorpe, Hardened properties of high-performance printing concrete, Cem. Concr. Res. 42 (3) (2012) 558–566, https://doi.org/10.1016/j.cemconres.2011.12.003.

[3] B. Zareiyan, B. Khoshnevis, Interlayer adhesion and strength of structures in contour crafting-effects of aggregate size, extrusion rate, and layer thickness, Autom. Constr. 81 (2017) 112–121, https://doi.org/10.1016/j.autcon.2017.06.013.

[4] Y.W. Tay, B. Panda, S.C. Paul, M.J. Tan, S.Z. Qian, K.F. Leong, C.K. Chua, Processing and properties of construction materials for 3D printing, Mater. Sci. Forum 861 (2016) 177–181, https://doi.org/10.4028/www.scientific.net/MSF.861.177.

[5] B. Khoshnevis, R. Dutton, Innovative rapid prototyping process makes large sized, smooth surfaced complex shapes in a wide variety of materials, Mater. Technol. 13 (2) (1998) 53–56, https://doi.org/10.1080/10667857.1998.11752766.

[6] G. Cesaretti, E. Dini, X. De Kestelier, V. Colla, L. Pambaguian, Building components for an outpost on the lunar soil by means of a novel 3D printing technology, Acta Astronaut. 93 (2014) 430–450, https://doi.org/10.1016/j.actaastro.2013.07.034.

[7] C. Gosselin, R. Duballet, P. Roux, N. Gaudilliere, J. Dirrenberger, P. Morel, Large-scale 3D printing of ultra-high performance concrete–a new processing route for architects and builders, Mater. Des. 100 (2016) 102–109, https://doi.org/10.1016/j.matdes.2016.03.097.

[8] S.J. Keating, J.C. Leland, L. Cai, N. Oxman, Toward site-specific and self-sufficient robotic fabrication on architectural scales, Science Robotics 2 (5) (2017) eaam8986, , https://doi.org/10.1126/scirobotics.aam8986.

[9] IAAC, Minibuilders, http://robots.iaac.net , Accessed date: 29 June 2018.

[10] T.C. Nguyen, Yes, that 3D-printed mansion is safe to live in, https://www.washingtonpost.com/news/innovations/wp/2015/02/05/yes-that-3d-printed-mansion-is-safe-to-live-in/?utm_term=.2112831042ec , Accessed date: 29 June 2018.

[11] S. Saunders, CyBe construction announces that 3D printing is complete for Dubai's R &Drone Laboratory, https://3dprint.com/176561/cybe-3d-printed-dubai-laboratory/ , Accessed date: 29 June 2018.

[12] R. Duballet, O. Baverel, J. Dirrenberger, Classification of building systems for concrete 3D printing, Autom. Constr. 83 (2017) 247–258, https://doi.org/10.1016/j.autcon.2017.08.018.

[13] J.G. Sanjayan, B. Nematollahi, M. Xia, T. Marchment, Effect of sur-face moisture on inter-layer strength of 3D printed concrete, Constr. Build. Mater. 172 (2018) 468–475, https://doi.org/10.1016/j. conbuildmat.2018.03.232.

[14] S.C. Paul, Y.W.D. Tay, B. Panda, M.J. Tan, Fresh and hardened properties of 3D printable cementitious materials for building and construction, Arch. Civ. Mech. Eng. 18 (1) (2018) 311–319, https://doi.org/10.1016/j.acme.2017.02.008.

[15] B. Panda, S.C. Paul, L.J. Hui, Y.W.D. Tay, M.J. Tan, Additive manufacturing of geopolymer for sustainable built environment, J. Clean. Prod. 167 (2017) 281–288, https://doi.org/10.1016/j.jclepro. 2017.08.165.

[16] B. Panda, S.C. Paul, M.J. Tan, Anisotropic mechanical performance of 3D printed fiber reinforced sustainable construction material, Mater. Lett. 209 (2017) 146–149, https://doi.org/10.1016/j.matlet.2017.07.123.

[17] M. Hambach, D. Volkmer, Properties of 3D-printed fiber-reinforced Portland cement paste, Cem. Concr. Compos. 79 (2017) 62–70, https://doi.org/10.1016/j.cemconcomp.2017.02.001.

[18] N. Khalil, G. Aouad, K. El Cheikh, S. Remond, Use of calcium sulfoaluminate cements for setting control of 3D-printing mortars, Constr. Build. Mater. 157 (2017) 382–391, https://doi.org/10.1016/j. conbuildmat.2017.09.109.

[19] P. Shakor, J. Sanjayan, A. Nazari, S. Nejadi, Modified 3D printed powder to cement-based material and mechanical properties of cement scaffold used in 3D printing, Constr. Build. Mater. 138 (2017) 398–409, https://doi.org/10.1016/j.conbuildmat.2017.02.037.

[20] A. Perrot, D. Rangeard, E. Courteille, 3D printing of earth-based materials: processing aspects, Constr. Build. Mater. 172 (2018) 670–676, https://doi.org/10.1016/j.conbuildmat.2018.04.017.

[21] A. Perrot, D. Rangeard, A. Pierre, Structural built-up of cement-based materials used for 3D-printing extrusion techniques, Mater. Struct. 49 (4) (2016) 1213–1220, https://doi.org/10.1617/s11527-015-0571-0.

[22] Y. Weng, M. Li, M.J. Tan, S. Qian, Design 3D printing cementitious materials via Fuller Thompson theory and Marson-Percy model, Constr. Build. Mater. 163 (2018) 600–610, https://doi.org/10.1016/j. conbuildmat.2017.12.112.

[23] R. Wolfs, F. Bos, T. Salet, Early age mechanical behaviour of 3D printed concrete: numerical modelling and experimental testing, Cem. Concr. Res. 106 (2018) 103–116, https://doi.org/10.1016/j.cemconres. 2018.02.001.

[24] R.P. Chhabra, J.F. Richardson, Non-Newtonian Flow and Applied Rheology: Engineering Applications, Butterworth-Heinemann, 978-0-7506-8532-0, 2011, https://doi.org/10.1016/B978-0-7506-8532-0.X0001-7.

[25] N. Roussel, A thixotropy model for fresh fluid concretes: theory, validation and applications, Cem. Concr. Res. 36 (10) (2006) 1797–1806, https://doi.org/10.1016/j.cemconres.2006.05.025.

[26] B. Khoshnevis, Automated construction by contour craftingrelated robotics and information technologies, Autom. Constr. 13 (1) (2004) 5–19, https://doi.org/10.1016/j.autcon.2003.08.012.

[27] B. Khoshnevis, G. Bekey, Automated construction using contour crafting–applications on earth and beyond, Nist Special Publication Sp, 2003, pp. 489–494, , https://doi.org/10.22260/ISARC2002/0076.

[28] B. Khoshnevis, D. Hwang, K.-T. Yao, Z. Yeh, Mega-scale fabrication by contour crafting, Int. J. Ind. Syst. Eng. 1 (3) (2006) 301–320, https://doi.org/10.1504/IJISE.2006.009791.

[29] T.T. Le, S.A. Austin, S. Lim, R.A. Buswell, A.G. Gibb, T. Thorpe, Mix design and fresh properties for high-performance printing concrete, Mater. Struct. 45 (8) (2012) 1221–1232, https://doi.org/10.1617/s11527-012-9828-z.

[30] S. Lim, R.A. Buswell, T.T. Le, S.A. Austin, A.G. Gibb, T. Thorpe, Developments in construction-scale additive manufacturing processes, Autom. Constr. 21 (2012) 262–268, https://doi.org/10.1016/j. autcon.2011.06.010.

[31] Cybe RC 3Dp, https://cybe.eu/3d-concrete-printers/#1520593810901-9a0ee625-b8d5 , Accessed date: 29 June 2018.

[32] T. Bailey, H. Durrant-Whyte, Simultaneous localization and mapping (SLAM): part II, IEEE Robot. Autom. Mag. 13 (3) (2006) 108–117, https://doi.org/10.1109/MRA.2006.1678144.

[33] M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (SLAM) problem, IEEE Trans.

Robot. Autom. 17 (3) (2001) 229–241, https://doi.org/10.1109/70.938381.
[34] S. Thrun, Simultaneous localization and mapping, Robotics and Cognitive Approaches to Spatial Mapping, Springer, 978-3-540-75386-5, 2007, pp. 13–41, , https://doi.org/10.1007/978-3-540-75388-9_3.
[35] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. Leonard, Past, present, and future of simultaneous localization and mapping: towards the robust-perception age, IEEE Trans. Robot. 32 (6) (2016) 13091332, https://doi.org/10.1109/TRO.2016.2624754.
[36] S.M. LaValle, Planning Algorithms, Cambridge University Press, 9780511546877, 2006, https://doi.org/10.1017/CBO9780511546877.
[37] S. Garrido-Jurado, R. Munñoz-Salinas, F.J. Madrid-Cuevas, M.J. Marın-Jimenez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recogn. 47 (6) (2014) 2280–2292, https://doi.org/10.1016/j.patcog.2014.01.005.
[38] Y. Zhou, H. Hu, Y. Liu, Z. Ding, Collision and deadlock avoidance in multirobot systems: a distributed approach, IEEE Trans. Syst. Man Cybern. Syst. 47 (7) (2017) 1712–1726, https://doi.org/10.1109/TSMC.2017.2670643.
[39] D. Claes, K. Tuyls, Multi robot collision avoidance in a shared workspace, Auton. Robot. (2018) 1–22, https://doi.org/10.1007/s10514-018-9726-5.
[40] E. Freund, H. Hoyer, Real-time pathfinding in multirobot systems including obstacle avoidance, Int. J. Robot. Res. 7 (1) (1988) 42–70, https://doi.org/10.1177/027836498800700104.
[41] D. Spensieri, J.S. Carlson, R. Bohlin, J. Kressin, J. Shi, Optimal robot placement for tasks execution, Procedia CIRP 44 (2016) 395–400, https://doi.org/10.1016/j.procir.2016.02.105.
[42] B. Kamrani, V. Berbyuk, D. Wappling, U. Stickelmann, X. Feng, Optimal robot placement using response surface method, Int. J. Adv. Manuf. Technol. 44 (1–2) (2009) 201–210, https://doi.org/10.1007/s00170-008-1824-7.
[43] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters, IEEE Trans. Robot. 23 (1) (2007) 34–46, https://doi.org/10.1109/TRO.2006.889486.
[44] D. Fox, S. Thrun, W. Burgard, F. Dellaert, Particle filters for mobile robot localization, Sequential Monte Carlo Methods in Practice, Springer, 2001, pp. 401–428, , https://doi.org/10.1007/978-1-4757-3437-9_19.
[45] Ubuntu, https://www.ubuntu.com , Accessed date: 29 June 2018.
[46] A. Koubaa, Robot Operating System (ROS), Springer, 0-470-23457-1, 2017, https://doi.org/10.1007/978-3-319-26054-9.
[47] P. Bosscher, R.L. Williams II, L.S. Bryson, D. Castro-Lacouture, Cable-suspended robotic contour crafting system, Autom. Constr. 17 (1) (2007) 45–55, https://doi.org/10.1016/j.autcon.2007.02.011.
[48] A. Ramirez-Serrano, R. Kuzyk, Modified mecanum wheels for traversing rough terrains, in: M. Bauer, J.L. Mauri, O. Dini (Eds.), The Sixth International Conference on Autonomic and Autonomous Systems, IEEE, Cancun, Mexico, 2010, pp. 97–103, , https://doi.org/10.1109/icas.2010.35.
[49] J.S. Russell, M.J. Skibniewski, J.A. Vanegas, Framework for construction robot fleet management system, J. Constr. Eng. Manag. 116 (3) (1990) 448–462, https://doi.org/10.1061/(ASCE)0733-9364(1990)116:3(448).
[50] M.J. Skibniewski, J.S. Russell, Construction robot fleet management system prototype, J. Comput. Civ. Eng. 5 (4) (1991) 444–463, https://doi.org/10.1061/(ASCE)0887-3801(1991)5:4(444).
[51] M.J. Skibniewski, R. Kunigahalli, J.S. Russell, Managing multiple construction robots with a computer, Autom. Constr. 2 (3) (1993) 199–216, https://doi.org/10.1016/0926-5805(93)90041-U.
[52] M.J. Skibniewski, K. Tamaki, Logistics support system for construction robotics implementation, J. Comput. Aided Civ. Infrastruct. Eng. 9 (1) (1994) 69–81, https://doi.org/10.1111/j.1467-8667.1994.tb00363.x.
[53] Nasa's centennial challenges: 3-D printed habitat challenge, https://www.nasa.gov/directorates/spacetech/centennial_challenges/3DPHab/index.html , Accessed date: 29 June 2018.

2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Departure and Conflict Management in Multi-Robot Path Coordination

Puttichai Lertkultanon[1], Jingyi Yang[2], Hung Pham[3], and Quang-Cuong Pham[3]

***Abstract*— This paper addresses the problem of multi-robot path coordination, considering specific features that arise in applications such as automatic aircraft taxiing or driver-less cars coordination. The first feature is departure events: when robots arrive at their destinations (e.g. the runway for take-off), they can be removed from the coordination diagram. The second feature is the "no-backward–movement" constraint: the robots can only move forward on their assigned paths. These features can interact to give rise to complex conflict situations, which existing planners are unable to solve in practical time. We propose a set of algorithms to efficiently account for these features and validate these algorithms on a realistic model of Charles de Gaulle airport.**

I. Introduction

Consider n robots whose tasks are to move from their initial positions $\boldsymbol{p}_i^{\text{start}}$ to their respective goal positions $\boldsymbol{p}_i^{\text{goal}}$, where $i \in \{1, 2, \ldots, n\}$. Suppose furthermore that their *paths* from $\boldsymbol{p}_i^{\text{start}}$ to $\boldsymbol{p}_i^{\text{goal}}$ are fixed and do not involve any collision with the environment. Finding the coordinated motions of the robots to move along their paths while avoiding mutual collisions is a classical problem in robotics, known as the *multi-robot path coordination* problem.

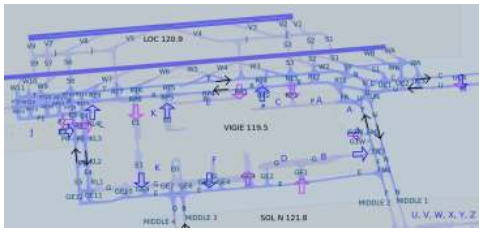
This problem arises in many contexts. In some multi-robot motion planning instances, the unconstrained problem (i.e. when the paths are not fixed) is intractable, in particular because of the high dimensionality of the problem itself, such that the only practical solution is to *decouple* it into two steps [1], [2], [3]: (i) find a path for each robot independently; (ii) solve the path coordination problem with the paths found in (i) being fixed. In some other applications, the robots have to move along predefined "tracks", e.g. driver-less cars on city streets, delivery vehicles in an automated warehouse, or automatic aircraft taxiing on airport tracks. Here, the constrains of "staying on the track" makes unconstrained motion planning inapplicable. A more natural solution consists of (i) finding the paths for each robot independently using, e.g. graph search; (ii) solving the path coordination problem with the paths found in step (i).

This work was partially supported by grant ATMRI:2014-R6-PHAM awarded by NTU and the Civil Aviation Authority of Singapore.

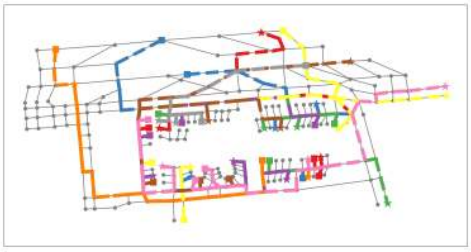
[1]Puttichai Lertkultanon was with the School of Mechanical and Aerospace Engineering, Nanyang Technological University (NTU), Singapore. He is now with Mujin, Inc., 1-1-9 Narihira, Sumida, Tokyo, Japan (email: L.Puttichai@gmail.com).

[2]Jingyi Yang was with the School of Mechanical and Aerospace Engineering, NTU, Singapore. She is now with Forming Technology Group, Singapore Institute of Manufacturing Technology, 2 Fusionopolis Way, Singapore (candice.yangjingyi@gmail.com).

[3]Hung Pham and Quang-Cuong Pham are with the School of Mechanical and Aerospace Engineering, Nanyang Technological University (NTU), Singapore (email: hungpham2511@gmail.com and cuong.pham@normalesup.org).



(a)



(b)

Fig. 1: (a) An actual map of the Charles de Gaulle (CDG) airport in Paris, France. (b) A simplified map of the airport used in the simulations presented in this paper. The figure includes paths of 25 aircraft, shown in different colors. Squares indicate start positions and stars indicate goal positions. A video of a coordination solution of 25 aircraft computed by the proposed planner can be found at `https://youtu.be/hfJeUKpeeD0`.

In this paper, we consider the multi-robot path coordination problem with automatic aircraft taxiing as an intended application (see Fig. 1 for an application scenario at Charles de Gaulle (CDG) airport in Paris, France). This application displays two specific features. The first feature is departure events: when robots arrive at their destinations (e.g. the runway for take-off), they can be removed from the coordination diagram, resulting in a change of the structure of the coordination problem, such as its dimensionality. The second feature is the "no-backward-movement" constraint: the robots can only move forward on their assigned paths. This constraint also arises in driver-less cars applications: while reversing a car is possible, it should be avoided in normal driving. Furthermore, these features can interact to give rise to complex conflict situations, which existing planners are unable to solve in practical times. The objective of this paper is to develop a set of algorithms to address (a) departure events, and (b) the "no-backward-movement" constraint in the multi-robot path coordination problem. As reported in Section V, these algorithms help reduce planning time of a sampling-based planner and endow the planner with the ability to solve problem instances with more aircraft. Consequently, they make coordinated motion

planning suitable for real-time taxiing applications.

The rest of the paper is organized as follows. In Section II, we briefly review the multi-robot motion planning problem and some of its applications. In Section III, we recall the main concepts of path coordination. In Section IV, we present our main contributions, namely, a set of new algorithms to handle departure and conflicts, which are specifically caused by the "no-backward-movement" constraint. In Section V, we present simulation results based on a realistic model of Charles de Gaulle airport. Finally, in Section VI, we discuss the advantages and drawbacks of our approach and sketch some future research directions.

II. Literature Review

Approaches to the problem of multi-robot motion planning can be classified as *centralized* or *decentralized*. In the centralized approach, a central computer receives position information, computes the motion plans, and sends the commands to all robots. On the other hand, in the decentralized approach, each robot computes its own movement based on the information at its disposal.

In the context of aircraft taxiing, the centralized approach is currently in use in most airports (air traffic controllers play the role of the central computer), which prompts us to focus on this approach in this paper. Within the centralized approach, one can further distinguish *decoupled* methods, which we have mentioned previously, from *non-decoupled* methods, where trajectories (paths *and* timings) are computed in a single step.

A. Non-Decoupled Methods

Applying classical robot motion planning methods, such as probabilistic roadmap (PRM) [4] or rapidly-exploring random tree (RRT) [5], directly to the product configuration space is highly inefficient. Similarly, graph search-based methods (see, e.g. [6]) will suffer from the curse of dimensionality. Therefore, most non-decoupled methods are found in the operation research literature, and are based in particular on Mixed Integer Linear Programming (MILP).

In [7], [8], Marín formulated the problem as a multi-commodity flow network model, which is then solved via MILP. In particular, a planning horizon is first defined and then discretized into a number of intervals with the horizon often lasts 30 min while the intervals is approximately 30 s. Then a spatial-time graph is constructed from the original spatial graph by creating multiple instances of each node corresponding to the number of time interval. The author then solved the resulting network flow problem with constraints via CPLEX. This formulation has the ability to consider general objective function. Note that recent developments on Lagrangian decomposition [8] yield significant improvements in terms of computation time.

A different formulation that also utilizes MILP was reported in [9]. Instead of time-discretization, binary variables were used to indicate the order with which aircraft pass through a given landmark. The authors reported that their implementation can plan motions for 30 aircraft in a reasonable amount of time (less than 100 s), but computation time increased significantly (more than 1000 s) for 50 aircraft.

B. Decoupled Methods

O'Donnell and Lozano-Pérez [1] were the first to introduce the idea of coordination diagram, which encodes the relative motions of the robots on their assigned paths (see Section III-A for more details). The authors then proposed to compute the obstacle regions in the coordination diagram explicitly via discretization and subsequently to find a collision-free path by a greedy algorithm.

In [3], Siméon *et al.* considered the problem of multiple mobile vehicles moving along straight lines or arcs. This particular path geometry allows them to efficiently compute the *bounding box* representation of the obstacle regions in the coordination diagram.

In [10], Peng and Akella considered the problem of time-optimal path coordination subject to velocity and acceleration constraints. The authors formulated the problem using Mixed Integer Nonlinear Programming and were able to compute sub-optimal but feasible coordinations.

In [2], Svetska and Overmars introduce the idea of roadmap coordination. Instead of moving on an assigned path, a robot can move within a roadmap. The coordination diagram then becomes a *composite roadmap*. This can be seen a generalization of both the non-decoupled and decoupled methods: if the roadmaps are infinitely dense, then roadmap coordination is equivalent to the former approach; if each roadmap contains only one path, then roadmap coordination is equivalent to the latter. Next, using roadmap coordination, it is possible to decompose the global coordination problem into several connected components.

More recently, in [11], Solovey *et al.* used RRT to find collision-free paths in the coordination diagram and obtained promising results in terms of computation time.

C. Specificity of Our Approach

Our approach is inscribed within the coordination diagram framework. In particular, as in [11], we use RRT to find collision-free paths in the coordination diagram. Our specificity consists of the departure and conflict management features. As we shall demonstrate, taking these features into account significantly improves the performance with respect to using a vanilla motion planner. Note that the development presented here can be easily combined with existing decoupled planning heuristics (e.g. roadmap coordination [2]).

In absence of standardized test cases, it is difficult to establish quantitative comparisons with the non-decoupled approaches based on MILP discussed previously. Indeed, the hardness of a motion planning instance not only depends on the number of aircraft, but more fundamentally, on the existence of "narrow passages". Our problem instances are associated with a large number of shared tracks and potential deadlocks, yielding particularly narrow passages in the coordination diagram, which are hard to overcome without the proposed conflict management.

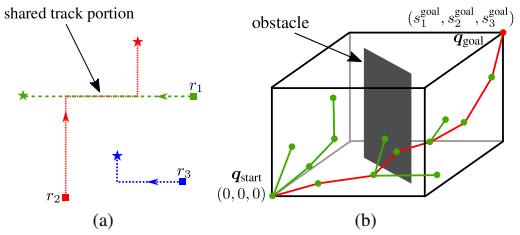


Fig. 2: (a) Three robot paths in the physical map. Squares are the start positions and Stars are the goal positions. Note that robots 1 and 2 share a common segment of their paths, on which they travel in opposite directions. (b) The corresponding coordination diagram. The obstacle induced by the shared path segment is depicted in dark gray. Note that for simplicity, the collision radius in this case was supposed to be very small. Otherwise, the obstacle would be thicker. The search tree is depicted in green and the solution path is highlighted in red.

III. Background: Multi-Robot Path Coordination without Backward Movements

A. Coordination Diagram

Consider n robots traveling on n given paths $P_1, P_2, \ldots, P_n$ with path lengths of $s_1^{\text{goal}}, s_2^{\text{goal}}, \ldots, s_n^{\text{goal}}$, respectively. The positions of the robots on their respective paths can be given by an n-tuple $\boldsymbol{q} = (s_1, s_2, \ldots, s_n)$, where $s_i \in [0, s_i^{\text{goal}}], 1 \leq i \leq n$. Thus, one can represent this vector, which we will refer to as a *configuration*, as a point in an n-orthotope $\mathscr{C} \coloneqq [0, s_1^{\text{goal}}] \times [0, s_2^{\text{goal}}] \times \ldots \times [0, s_n^{\text{goal}}]$, also called the *coordination diagram* (CD) [1], [2], [3].

As two paths P_i and P_j may intersect or even share some track segments, robots i and j may collide with each other while traversing their paths. For simplicity, robots are assumed to be identical and are represented as disks of radius $r/2$. One can thus define the *obstacle set* in $\mathscr{C}$ as

$$\mathscr{O} = \{\boldsymbol{q} \,|\, \exists i, j, \|\boldsymbol{p}_i(s_i) - \boldsymbol{p}_j(s_j)\|_2 \leq r\}, \tag{1}$$

where $\boldsymbol{p}_i(s)$ is the 2D physical position of the robot i when it has traveled a distance s along its path.

The problem of path coordination—finding the motions for all robots along their paths so that they (i) start from the origin of their paths; (ii) eventually reach the end of their paths; and (iii) never collide with one another—can now be cast as a classical motion planning problem: find a collision-free continuous path between $(0, 0, \ldots, 0)$ and $(s_1^{\text{goal}}, s_2^{\text{goal}}, \ldots, s_n^{\text{goal}})$ in the CD. Fig. 2 shows an example of coordination diagram and a search tree constructed for a problem with three robots.

The classical motion planning problem can be solved by any existing geometric motion planning algorithms [12]. Here we use a rapidly-exploring random tree (RRT) planner [5]. Briefly, RRT iteratively grows a tree rooted at the start configuration $\boldsymbol{q}_{\text{start}} \coloneqq (0, 0, \ldots, 0)$. At each planning iteration, a random configuration $\boldsymbol{q}_{\text{rand}}$ is sampled within the free space $\mathscr{C} \setminus \mathscr{O}$. Next, one determines amongst the existing tree vertices, the vertex $\boldsymbol{q}_{\text{near}}$ that is the nearest neighbor to $\boldsymbol{q}_{\text{rand}}$, according to some distance function dist. Finally, one tries extending the tree from $\boldsymbol{q}_{\text{near}}$ towards $\boldsymbol{q}_{\text{rand}}$, creating thereby a new tree vertex $\boldsymbol{q}_{\text{new}}$. The algorithm terminates when either the time limit is reached or the goal configuration $\boldsymbol{q}_{\text{goal}} \coloneqq (s_1^{\text{goal}}, s_2^{\text{goal}}, \ldots, s_n^{\text{goal}})$ can be connected to the tree. For more details and variations, the reader is referred to [5].

Collision Checking: When extending the tree from $\boldsymbol{q}_{\text{near}}$ towards $\boldsymbol{q}_{\text{rand}}$, one needs to check if the segment $(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{rand}})$ [1] is collision-free. For this, one checks whether all discretized configurations $\boldsymbol{q}$ along the segment are collision-free.

Consider a configuration $\boldsymbol{q} = (s_1, s_2, \ldots, s_n)$. To check whether $\boldsymbol{q}$ is collision-free, one puts all the robots to the physical positions defined by the respective $s_i, i \in \{1, 2, \ldots, n\}$. Then one checks for all pairs (i, j) whether the robots i and j are in mutual collision, according to Eq. (1). Note that the obstacle set is thus not necessarily explicitly constructed in the coordination diagram.

B. Motion Planning without Backward Movements

Consider a straight path from $\boldsymbol{q} = (s_1, s_2, \ldots, s_n)$ to $\boldsymbol{q}' = (s_1', s_2', \ldots, s_n')$ in $\mathscr{C}$. The "no-backward-movement" constraint is equivalent to the monotonicity of the path. Therefore, the path satisfies the "no-backward-movement" constraint *if and only if* $\boldsymbol{q} \preceq \boldsymbol{q}'$, where $\preceq$ is a component-wise inequality between two vectors.

To enforce this constraint in RRT, one should attempt extension from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{rand}}$ only when $\boldsymbol{q}_{\text{near}} \preceq \boldsymbol{q}_{\text{rand}}$. A simple way to implement this filter is to modify the distance function (used in the nearest neighbor search) as

$$\text{dist}(\boldsymbol{q} \to \boldsymbol{q}') = \begin{cases} \|\boldsymbol{q} - \boldsymbol{q}'\| & \text{if} \quad \boldsymbol{q} \preceq \boldsymbol{q}' \\ +\infty & \text{otherwise} \end{cases} \tag{2}$$

such that any $\boldsymbol{q} \not\preceq \boldsymbol{q}_{\text{rand}}$ will never be selected as the nearest neighbor of $\boldsymbol{q}_{\text{rand}}$.

IV. Departure and Conflict Management

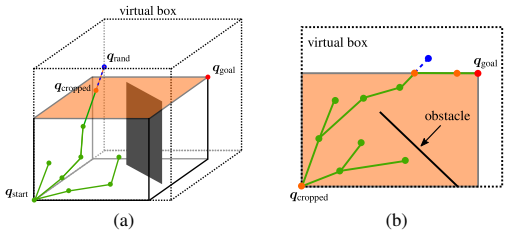
A. Departure Management

We say that a robot *departs* from the CD when it reaches its goal position. This terminology is in line with the aircraft taxiing context, where aircraft indeed depart (take off) when they reach their goal positions (the runway) or wheel-locked in the gate where it would not block the way of other aircraft. Once a robot has departed, it "disappears" and is no longer taken into account in collision checks, thereby simplifying the motion planning problem.

In the CD $\mathscr{C}$, departure of a robot corresponds to the search tree reaching a configuration $\boldsymbol{q}$ on an $n-1$-dimensional *face* of $\mathscr{C}$. When this happens, we continue the planning *on the respective face of* $\mathscr{C}$. This can be seen as initiating a new search tree, in $n-1$-dimensional space, rooted at the intersection point of the previous tree and the face. When another robot reaches its goal position, we continue the same process, i.e. planning in the $n-2$-dimensional space and so on.

Virtual Box: If one samples new configurations within the CD $\mathscr{C}$, one can never reach the faces of $\mathscr{C}$ since the faces have measure zero relative to $\mathscr{C}$. Therefore, we propose to sample new configurations within a *larger* n-orthotope $\overline{\mathscr{C}} \coloneqq$

[1] In case $\text{dist}(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{rand}}) > \varepsilon$, where ε is the maximum extension distance, one considers instead the segment $(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{close}})$, where $\boldsymbol{q}_{\text{close}}$ is the configuration on the segment $(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{rand}})$ at distance ε from $\boldsymbol{q}_{\text{near}}$.



(a) (b)

Fig. 3: Departure management. (a) At iteration k, a newly sampled configuration $\boldsymbol{q}_{\text{rand}}$ falls outside $\mathscr{C}$. The interpolated path is thresholded at $\boldsymbol{q}_{\text{cropped}}$, which lies on a face of $\mathscr{C}$ (shaded in orange). (b) From iteration $k+1$ on, the algorithm continues planning *on the face* until the search tree reaches $\boldsymbol{q}_{\text{goal}}$.

$[0, \overline{s_1}] \times \ldots \times [0, \overline{s_n}]$, where $\overline{s_i} \geq s_i^{\text{goal}}, i \in \{1, 2, \ldots, n\}$. We call $\overline{\mathscr{C}}$ the *virtual box*.

When a sampled configuration $\boldsymbol{q}_{\text{rand}}$ is in $\overline{\mathscr{C}} \setminus \mathscr{C}$, we first find $\boldsymbol{q}_{\text{near}}$ as the tree vertex that is closest to $\boldsymbol{q}_{\text{rand}}$ as previously. Next, when extending the tree from $\boldsymbol{q}_{\text{near}}$ towards $\boldsymbol{q}_{\text{rand}}$, we stop at $\boldsymbol{q}_{\text{cropped}}$, defined as the intersection of the segment $(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{rand}})$ [2] with the boundary of $\mathscr{C}$, see Fig. 3.

The size of the virtual box, i.e. the values $\overline{s_i}$, also affect the performance of the planner. The greater the values $\overline{s_i}$, the more the tree is attracted towards the faces of $\mathscr{C}$ and the relative value $\overline{s_i}/s_i^{\text{goal}}$ determine the attractiveness of the face i. These attractiveness ratio can also be seen as priority given to robots as the greater ratio, the more likely the robot will reach its goal first.

In our implementation, we chose $\overline{s_1} = \overline{s_2} = \ldots = \overline{s_n} = \gamma \max_i(s_i^{\text{goal}})$, where $\gamma \geq 1$ is a design parameter. Choosing an n-orthotope with equal side lengths (i.e. a cuboid) as the virtual box favors the robots with shorter path lengths since the attractiveness ratio increases with shorter path lengths. Thus, the algorithm will tend to make those robots reach their goals first and "clear up the floor" more quickly, simplifying thereby the motion planning. Regarding the value γ, we found through extensive simulations that $\gamma = 1.2$ yields the best result.

B. Conflict Management

An *elementary conflict* happens when two robots, traversing their own assigned paths, have potential to collide with each other. This, in turn, happens when their paths *intersect*, either at a point or a long a segment. There are three types of elementary conflicts:

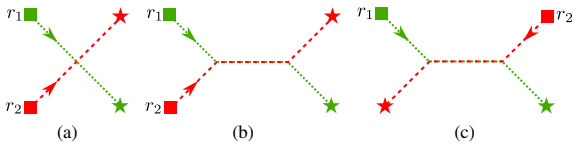
Junction (J): two robot paths cross each other at a point;
Shared (S): two robot paths share a path segment where both robots travel in the same direction; and
Deadlock (D): two robot paths share a path segment where the two robots travel in opposite directions.

The scenarios of elementary conflicts are depicted in Fig. 4.

A sampling-based planner, equipped with a collision checker[3], is able to explore the configuration space without

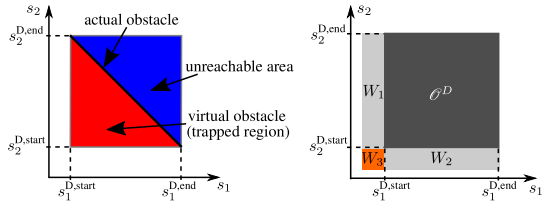
[2]Note that for simplicity of the discussion, here we ignore the maximum extension distance ε. This maximum extension distance must also be taken into account in the actual implementation.

[3]A collision checker is a function that takes a configuration as its input and return a Boolean value indicating if the input configuration is in collision.



(a) (b) (c)

Fig. 4: Three types of elementary conflicts: (a) Junction (type-J); (b) Shared (type-S); (c) Deadlock (type-D).



(a) (b)

Fig. 5: Deadlock obstacle $\mathscr{O}^D$. (a) The black diagonal line represents configurations where actual collisions occur. The red region is a trapped region where there exists no collision-free and monotonically increasing path that connects a configuration inside to the goal. (b) The waiting areas W_1, W_2, and W_3. When the search tree is extended from a configuration inside a waiting area, one robot should be made temporarily inactive (i.e. waiting) until the other robot has exited the deadlock path segment.

the need to explicitly computing the obstacle set, which is computationally expensive. However, we found out that:

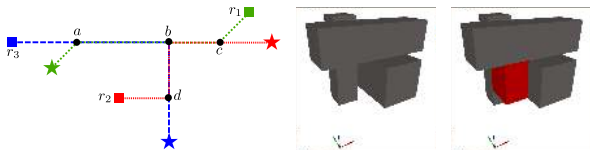
- The presence of narrow passages—regions in $\mathscr{C}$ with relatively small volumes that when removed, increase the number of connected component in $\mathscr{C}$—greatly affects the performance of the planner. While it is possible to use some heuristics to alleviate the problems arisen from narrow passages (see, e.g. [13]), they often require some assumptions on the obstacle set $\mathscr{O}$. Therefore, it is better to have at least partial knowledge of $\mathscr{O}$ in order to devise policies to overcome narrow passages.
- The obstacles in $\mathscr{C}$ that are induced by elementary conflicts can be closely described by simple geometric shapes and do not require extensive computational resource to compute and store them.

Therefore, our approach is to compute the obstacle sets induced by conflicts and to devise policies to overcome those obstacles.

1) Representing Conflicts in $\mathscr{C}$: For simplicity of discussion, we suppose that the collision radius r is very small. In practice, the radius r can be taken into account by padding the obstacle set by the value r.

Consider when there are two robots, r_1 and r_2, in the scene. The paths of the two robots are parameterized by parameters $s_1 \in [0, s_1^{\text{goal}}]$ and $s_2 \in [0, s_2^{\text{goal}}]$, respectively.
Junction: The paths intersect at $(s_1, s_2) = (s_1^J, s_2^J)$. The obstacle set $\mathscr{O}^J$ is described as $\mathscr{O}^J = \{(s_1^J, s_2^J)\}$.
Shared: Suppose the robot r_1 is in the shared segment when $s_1 \in I_1^S := [s_1^{\text{S,start}}, s_1^{\text{S,end}}]$ and the robot r_2 is in the shared segment when $s_2 \in I_2^S := [s_2^{\text{S,start}}, s_2^{\text{S,end}}]$. The obstacle set is described as $\mathscr{O}^S = \{(s_1, s_2) \,|\, s_1 \in I_1^S, s_2 \in I_2^S, s_1 = s_2 + (s_1^{\text{S,start}} - s_2^{\text{S,start}})\}$[4], which is a straight line connecting

[4]Note that since all the paths are parameterized by their own path lengths, the intervals I_1 and I_2 have the same length. This implies $s_1^{\text{S,start}} - s_1^{\text{S,start}}$ and $s_1^{\text{S,end}} - s_1^{\text{S,end}}$, and so on.



(a) (b) (c)

Fig. 6: Complex deadlocks. (a) The situation consists of three robots. (b) There are three elementary deadlock obstacles in $\mathscr{C}$, depicted as gray rectangular prisms (not to scale). (c) The three elementary deadlock obstacles induce a complex deadlock shown in red.

the points $(s_1^{\text{S,start}}, s_2^{\text{S,start}})$ and $(s_1^{\text{S,end}}, s_2^{\text{S,end}})$ in $\mathscr{C}$.

Deadlock: Suppose the robot r_1 is in the shared segment when $s_1 \in I_1^D := [s_1^{\text{D,start}}, s_1^{\text{D,end}}]$ and the robot r_2 is in the shared segment when $s_2 \in I_2^D := [s_2^{\text{D,start}}, s_2^{\text{D,end}}]$. The obstacle set is described as $\mathscr{O}^{D'} = \{(s_1, s_2) \,|\, s_1 \in I_1^D, s_2 \in I_2^D, s_1 + s_2 = (s_1^{\text{D,start}} + s_2^{\text{D,start}})\}$, which is a straight line connecting $(s_1^{\text{S,start}}, s_2^{\text{S,end}})$ and $(s_1^{\text{S,end}}, s_2^{\text{S,start}})$ in $\mathscr{C}$.

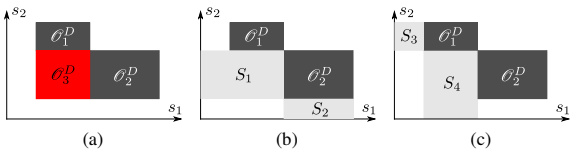
Note, however, that the deadlock obstacle $\mathscr{O}^{D'}$ also creates a *virtual obstacle* or a *trapped region* (the red region in Fig. 5). For any configuration (s_1, s_2), although they are collision-free, there exists no monotonically increasing path that connects (s_1, s_2) to $(s_1^{\text{goal}}, s_2^{\text{goal}})$. Therefore, the search tree should not extend itself to any configuration inside any trapped region. Also, the area *above* $\mathscr{O}^{D'}$ (the blue region in Fig. 5) is not reachable due to the "no-backward-movement" constraint. Therefore, we propose to describe a deadlock obstacle as $\mathscr{O}^D = I_1^D \times I_2^D$, which is the Cartesian product of the two intervals.

When there are n robots in the scene, the obstacles are simply protrusions of the previously discussed obstacle sets into the new dimensions. For example, from the scenario in Fig. 2(a), the deadlock obstacle is a rectangular prism $I_1^D \times I_2^D \times [0, s_3^{\text{goal}}]$.

2) Resolving Conflicts: Here we only discuss a policy to overcome deadlock obstacles since they have the most volume compared to obstacles of other types hence highest potential to create narrow passages.

Since one robot can be inside a deadlock path segment at a time, when one robot, say r_1, has already entered the deadlock, the other, say r_2, needs to *wait* until r_1 robot has come out. Also, r_2 should be waiting only when it is near the entrance of the deadlock segment. Geometrically, this policy leads us to define waiting areas W_1, W_2, and W_3 in $\mathscr{C}$, as depicted in Fig. 5(b). When the search tree is extended from a configuration inside a waiting area, one robot should be made temporarily inactive to prevent future collision. If the tree is extended from $\boldsymbol{q}_{\text{near}} \in W_1$, the robot r_1 should be made waiting, i.e. the tree extension will go *vertically*. In case of W_2, the robot r_2 should be waiting, while in case of W_3, the user has freedom to choose the waiting robot. Note that the dimension of the waiting area can be adjusted to suit different applications.

3) Complex Deadlocks: Apart from elementary deadlock obstacles, as discussed previously, there exists *complex deadlocks*, i.e. ones that involve more than two robots. Note that this complex deadlock is *not* equivalent to multiple elementary deadlocks since it is not a subset of any combination of



(a) (b) (c)

Fig. 7: Complex deadlock computation. (a) There are two elementary obstacles $\mathscr{O}_1^D$ and $\mathscr{O}_1^D$, which are connected at their corners. These elementary obstacles induce a complex deadlock $\mathscr{O}_3^D$, shaded in red. To compute $\mathscr{O}_3^D$, we compute shadows of $\mathscr{O}_1^D$ (see (b)) and shadows of $\mathscr{O}_1^D$ (see (c)). Then the complex deadlock is the non-empty intersection between shadows of different directions. In this case, $\mathscr{O}_3^D = S_1 \cap S_4$.

elementary obstacles.

To see this, consider a situation depicted in Fig. 6(a) in which there are three robots. There is a deadlock between every possible pair of robots. Each of the conflicts induces a deadlock obstacle in $\mathscr{C}$ that is a 3-orthotope. However, in this case, the arrangement of the three 3-orthotopes is such that it induces a new trapped region not inside any existing elementary deadlock obstacles (red region in Fig. 6(c)). We call this trapped region a *complex deadlock* as it is induced by multiple elementary obstacles. To prevent the search tree from being stuck in a complex deadlock, we present an algorithm to compute a complex deadlock obstacle induced by a cluster of elementary deadlock obstacles.

Consider an exemplary 2D coordination diagram with two deadlock obstacles, $\mathscr{O}_1^D$ and $\mathscr{O}_2^D$, connected at their corners, as depicted in Fig. 7(a). In this case, there exists a complex deadlock $\mathscr{O}_3^D$, shaded in red. To compute $\mathscr{O}_3^D$, we need to compute *shadows* produced by each elementary obstacle. Then a complex deadlock will be an intersection of shadows. More precisely, suppose there is a light source placed at $+\infty$ of s_1-axis of $\mathscr{C}$. Then $\mathscr{O}_1^D$ will produce a shadow S_1 and $\mathscr{O}_2^D$ will produce a shadow S_3 (see Fig. 7). Configurations in these shadows can be seen as being *blocked* in s_1-direction. Suppose there is another light source placed at $+\infty$ of s_2-axis then we compute shadows S_2 and S_4. Similarly, configurations in S_2 or S_4 can be seen as being blocked in s_2-direction. Finally, the complex deadlock is obtained as an intersection of shadows from different direction. This is because a deadlock is, by definition, a region that is blocked in every direction. In this example, $\mathscr{O}_3^D = S_1 \cap S_4$.

Note that the proposed procedure of computing complex deadlocks can be easily generalized to handle complex deadlocks or any dimension. For example, consider an obstacle described by a Cartesian product $[s_1', s_1''] \times [s_2', s_2''] \times \ldots \times [s_k'.s_k'']$. The shadow of this obstacle in s_1-direction is simply described as $[0, s_1'] \times [s_2', s_2''] \times \ldots \times [s_k'.s_k'']$.

V. Simulations

A. Planning Pipeline

We propose the following pipeline to address multi-robot path planning problems:

Step 1 Plan robot paths in the physical map independently using, e.g. a graph search algorithm;

Step 2 Find a collision-free path in the coordination diagram using the set of algorithms developed previously;

Step 3 Post-process the path.

In the post-processing step, one can first use shortcutting algorithms [14], [15] to smoothen the path in the CD. Then one can *time-parameterize* the physical paths, i.e. impose velocity and acceleration constraints, by using algorithms such as Time-Optimal Path Parameterization (TOPP) [16] [5]. If the commands are to be transmitted by Air Traffic Controllers, one can also impose the switch times (the time duration between two acceleration switches) to be always larger than some predefined minimum value [15].

B. Simulation Results

Here we compare performance of four variants of the RRT planner [5] in coordination scenarios on the CDG airport (see Fig. 1) involving 2 to 25 aircraft. A video of a coordination solution of a problem with 25 aircraft can be found at `https://youtu.be/hfJeUKpeeD0`. All the simulations were run on a 2.2GHz laptop running Ubuntu.

RRT Variants: The four planners are as follows:

1) Vanilla RRT: This planner is without handling of departure events (virtual box) and conflict management. After each successful extension of the search tree, the planner will attempt to connect the newly added tree vertex to the goal configuration via a straight path with a probability p_{bias}.
2) RRT-DM: This is an RRT planner with only the **D**eparture **M**anagement feature, i.e. the virtual box heuristic.
3) RRT-CM: This is an RRT planner with only the **C**onflict **M**anagement feature. Since the virtual box heuristic is not used, there is also a connection attempt after each successful tree extension as in the vanilla RRT.
4) RRT-DCM: This is an RRT planner with both the departure management and conflict management features.

In the simulations, all variants used the same maximum extension distance (ε). The probability p_{bias} (for Vanilla RRT and RRT-CM) was set to 0.2.

Problem Settings: Let $2 \leq n \leq 25$ be the number of robots in the scene. For each n, we first randomly sampled n pairs of start and goal positions. Then for each planner, we ran simulations 50 times on each of the 24 sampled problem instances. In each run, the time limit was set to 60 s.

Results are reported in Fig. 8. While RRT-DCM found a solution in every run, the success rates of the other three planners decrease rapidly with n. At large n's, those planners only occasionally get *lucky* enough to find solutions. From this sufficiently clear trend, we did not run simulations for Vanilla RRT, RRT-DM, and RRT-CM for $n > 15$.

Interestingly, RRT-DM performed better than Vanilla RRT and RRT-CM. This is because in a multi-robot path coordination scenario, allowing robots to reach their goals at different times greatly increases the solution space, hence the high success rate. However, since a new search tree is initiated when a robot reaches its goal, the lack of conflict management feature can result in the new tree being trapped completely in a deadlock. Similarly, although the conflict management feature is useful, not explicitly allowing robots to reach their goals at different times is still too restrictive that although the search tree did not get trapped, the planner required too much time to find a solution.

One can see that while the presented two features—departure handling and conflict management—are not much beneficial being used separately, they are complementary. Therefore, integrating both of them into a planner significantly improved the performance as illustrated in simulations.

Note also that although previous works such as [17] presented simulations with 240 aircraft, the number of aircraft was cumulative across an interval of three hours. All snapshots presented in [17] showed less than 20 aircraft in the airport. By contrast, the simulations presented here were such that all aircraft were presented in the scene.

[5]Note that the time-parameterizability also depends on the geometry of the path in the CD (the readers are referred to [16] for more details).

VI. Discussion and Conclusion

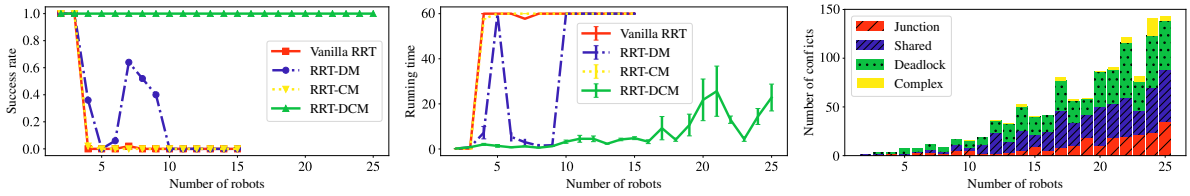
A. Discussion

Arrival Management: This feature cannot be directly integrated into the presented approach since the planning in the coordination diagram does *not* include timing. The planned path only encodes information of *relative positions* of robots/aircraft. It is thus not possible to include new aircraft into the planner while in the middle of the planning process. However, the simulation results presented previously suggest that it can be possible to use the approach "plan-execute-replan". That is, it is feasible to assume that a coordinated path is planned and the execution starts before a new aircraft arrives. Thus, when a new aircraft arrives, one can *replan* the coordinated path by using the current configuration as the root of the new search tree.

Complex Deadlocks: The existence of complex deadlocks has been already been discussed in the literature (see, e.g. [1]). Similar ideas were also explored in literature on concurrency control of databases [18], [19]. However, previous works only discussed these deadlocks in 2D cases. Furthermore, a complex deadlock was also thought to be the *SW-hull* [20] of other existing obstacles [1]. Upon further investigation, we found out that the complex deadlocks are equivalent to SW-hulls only in two-dimensional cases. To our knowledge, this paper is the first to develop an algorithm to explicitly address high-dimensional complex deadlocks.

The presented approach, however, comes with high combinatorial complexity. To compute a complex deadlock induced by a cluster of m elementary deadlocks, we compute exactly two shadows produced by each obstacle. In the worst case, we need to examine all possible cases 2^m of shadow intersection (since in each case, we can select one out of two shadows). Therefore, the worst case complexity is $O(2^m)$. It is possible to reduce the computation time by using, for example, a graph search algorithm that takes into account path overlapping, hence reducing m. However, to derive a more efficient complex deadlock detection algorithm, we need to gain more understanding of how complex deadlocks are formed, their mathematical properties, etc.

Completeness: The RRT planner itself is probabilistically complete [5], meaning that if there exists a solution, it will



(a) (b) (c)

Fig. 8: Experimental results. (a) The success rates of all planners as a function of the number of involving aircraft. (b) The average running time over 50 runs of all planners as a function of the number of aircraft. The vertical bars indicate the respective standard deviation. In case a planner failed in all 50 runs, the running time of that problem instance is shown as the time limit, 60 s. (c) The number of conflicts of each type in the sampled problem instances used in the simulations.

be found with probability one as the number of iterations increases. However, the planner presented in this paper *may* not be probabilistically complete. One factor is that the current mechanism of departure management reduces the dimension of the coordination diagram immediately after a robot has reached its goal[6]. This mechanism implicitly assumes that the configuration $\boldsymbol{q}_{\text{cropped}}$ is part of some solutions, which we currently have no guarantee. Further study should also focus on this completeness issue.

Optimality: Although optimal sampling-based planners such as [21] can be used instead of RRT, further analysis is needed in order to verify whether the optimality of such planners is retained under the presence of the no-backward-movement constraint, which limits the connectivity between a tree vertex and its neighbors.

B. Conclusion

In this paper, we have developed a set of algorithms to deal with departure events and conflicts (mainly caused by the "no-backward-movement" constraint) in the classical multi-robot path coordination problem. The virtual box heuristic, when integrated into a planner, allows robots to reach their goals at different times while the conflict management feature helps prevent the search tree from being trapped in a deadlock situation. Integrating the presented two features in a multi-robot coordination planner greatly improves the performance, as illustrated in simulations. Future research directions include, but not limited to, 1) further understanding of deadlock formation and how to reduce complexity of complex deadlock computation; and 2) analyzing the completeness property of the planner.

ACKNOWLEDGMENT

We thank our colleagues from ENAC, Railane Benhacene and Mathieu Cousy, for directing us towards the aircraft taxiing problem as well as for providing the data for Charles de Gaulle airport.

REFERENCES

[1] P. A. O'Donnell and T. Lozano-Pérez, "Deadlock-free and collision-free coordination of two robot manipulators," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, 1989, pp. 484–489.

[6]One trivial way to overcome this issue, which comes with the cost of higher computational complexity, is to retain the whole planning tree even when some robots have already reached their goals. This allows the planner to still consider different orders of robots reaching their goals.

[2] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and Autonomous Systems*, vol. 23, no. 3, pp. 125–152, April 1998.
[3] T. Siméon, S. Leroy, and J.-P. Laumond, "Path coordination for multiple mobile robots: a resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.
[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
[6] H. Ma, J. Li, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS),2017*, 2017, pp. 837–845.
[7] Á. G. Marín, "Airport management: Taxi planning," in *Annals of Operations Research*, vol. 143, no. 1, 2006, pp. 191–202.
[8] ——, "Airport taxi planning: Lagrangian decomposition," *Journal of Advanced Transportation*, vol. 47, no. 4, pp. 461–474, 2013.
[9] M. J. S. J. W. Smeltink, "An optimisation model for airport taxi scheduling," 2004.
[10] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.
[11] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," in *Springer Tracts in Advanced Robotics*, vol. 107, 2015, pp. 591–607.
[12] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.
[13] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.
[14] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.
[15] P. Lertkultanon and Q.-C. Pham, "Time-optimal parabolic interpolation with velocity, acceleration, and minimum-switch-time constraints," *Advanced Robotics*, vol. 30, no. 17–18, pp. 1095–1110, 2016.
[16] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, pp. 1533–1540, 2014.
[17] G. L. Clare and A. G. Richards, "Optimization of taxiway routing and runway scheduling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1000–1013, 2011.
[18] M. Yannakakis, C. H. Papadimitriou, and H. T. Kung, "Locking policies: Safety and freedom from deadlock," in *Foundations of Computer Science, 1979., 20th Annual Symposium on*, 1979, pp. 286–297.
[19] W. Lipski and C. H. Papadimitriou, "A fast algorithm for testing for safety and detecting deadlocks in locked transaction systems," *Journal of Algorithms*, vol. 2, no. 3, pp. 211–226, 1981.
[20] F. P. Preparata and M. Shamos, *Computational Geometry*. Springer-Verlag New York, 1985.
[21] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

Bibliography

[1] António Pedro Aguiar and João P Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[2] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[3] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 1996. ISSN 0166218X. doi: 10.1016/0166-218X(95)00026-N.

[4] Reinhold Behringer, Sundar Sundareswaran, Brian Gregory, Richard Elsley, Bob Addison, Wayne Guthmiller, Robert Daily, and David Bevly. The DARPA grand challenge-development of an autonomous vehicle. In *IEEE Symposium on Intelligent Vehicles*, pages 14–17, 2004.

[5] Richard Bellman. *Dynamic programming*. Courier Corporation, 2013.

[6] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[7] Aharon Ben-Tal and Arkadi Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001.

[8] Dimitri P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[9] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific Belmont, MA, 2007.

[10] Dimitri P. Bertsekas and I.B. Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7(2):233–247, mar 1971. ISSN 00051098. doi: 10.1016/0005-1098(71)90066-5. URL http://linkinghub.elsevier.com/retrieve/pii/0005109871900665.

[11] John T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998. ISSN 0731-5090. doi: 10.2514/2.4231. URL http://arc.aiaa.org/doi/abs/10.2514/2.4231.

[12] Akilan Bharathi and Jingyan Dong. A Smooth Trajectory Generation Algorithm for Addressing Higher-Order Dynamic Constraints in Nanopositioning Systems. *Procedia Manufacturing*, 1:216–225, 2015.

[13] Corrado Guarino Lo Bianco. Minimum-jerk velocity planning for mobile robot applications. *IEEE Transactions on Robotics*, 29(5):1317–1326, 2013.

[14] Corrado Guarino Lo Bianco and Oscar Gerelli. Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints. *IEEE Transactions on Robotics*, 27(6):1144–1152, 2011. ISSN 15523098. doi: 10.1109/TRO.2011.2162268.

[15] Corrado Guarino Lo Bianco and Fabio Ghilardelli. Real-time planner in the operational space for the automatic handling of kinematic constraints. *IEEE Transactions on Automation Science and Engineering*, 11(3):730–739, 2014. ISSN 15455955. doi: 10.1109/TASE.2014.2310813.

[16] James E. Bobrow, Steven 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.

[17] James E. Bobrow, J. Michael McCarthy, and V. K. Chu. Minimum-time trajectories for two robots holding the same workpiece. In *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 3102–3107. IEEE, 1990.

[18] Martin Böck and Andreas Kugi. Real-time nonlinear model predictive path-following control of a laboratory tower crane. *IEEE Transactions on Control Systems Technology*, 22(4):1461–1473, 2014.

[19] Tom Bonkenburg. Robotics in logistics. Technical report, DHL Customer Solutions & Innovation, 2016. URL http://www.dhl.com/content/dam/downloads/g0/about_us/logistics_insights/dhl_trendreport_robotics.pdf.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[21] Tye Michael Brady and Ethan Zane Evans. Autonomous ground vehicles based at delivery locations, Jan 2018. US Patent App. 15/218,943.

[22] Timothy Bretl and Sanjay Lall. Testing static equilibrium for legged robots. *IEEE Transactions on Robotics*, 24(4):794–807, 2008.

[23] Stéphane Caron and Quang-Cuong Pham. When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC. *arXiv preprint arXiv:1609.04600*, 2016.

[24] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics. *Robotics: Science and System*, 2015.

[25] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5107–5112. IEEE, 2015.

[26] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Zmp support areas for multicontact mobility under frictional constraints. *IEEE Transactions on Robotics*, 33(1):67–80, 2017.

[27] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.

[28] Daniela Constantinescu and Elizabeth A. Croft. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of robotic systems*, 17(5):233–249, 2000.

[29] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, Massachusetts London, Mcgraw-hill Book Company, and Boston Burr Ridge. *Introduction to Algorithms, Second Edition*, volume 7. The MIT Press, 2001. ISBN 0262032937.

[30] Nikolaus Correll, Kostas E. Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M. Romano, and Peter R. Wurman. Analysis and Observations from the First Amazon Picking Challenge. *IEEE Transactions on Automation Science and Engineering*, 2016. ISSN 1545-5955. doi: 10.1109/TASE.2016.2600527. URL http://arxiv.org/abs/1601.05484.

[31] Gábor Csorvási, Ákos Nagy, and István Vajk. Near Time-Optimal Path Tracking Method for Waiter Motion Problem. *IFAC-PapersOnLine*, 50 (1):4929–4934, jul 2017. ISSN 24058963. doi: 10.1016/j.ifacol.2017.08.749. URL https://www.sciencedirect.com/science/article/pii/S2405896317311965.

[32] Ola Dahl. Path-constrained robot control with limited torques-experimental evaluation. *IEEE transactions on robotics and automation*, 10(5):658–669, 1994.

[33] Ola Dahl and Lars Nielsen. Torque-limited path following by online trajectory time scaling. *Robotics and Automation, IEEE Transactions on*, 6(5):554–561, 1990.

[34] Frederik Debrouwere, Wannes Van Loock, Goele Pipeleers, Quoc Tran Dinh, Moritz Diehl, Joris De Schutter, and Jan Swevers. Time-optimal path following for robots with convex-concave constraints using sequential convex programming. *IEEE Transactions on Robotics*, 29(6):1485–1495, 2013. ISSN 15523098. doi: 10.1109/TRO.2013.2277565.

[35] Rosen Diankov and James Kuffner. OpenRAVE : A Planning Architecture for Autonomous Robotics. *Robotics*, pages –34, 2008. URL http://www.ri.cmu.edu/pub_files/pub4/diankov_rosen_2008_2/diankov_rosen_2008_2.pdf.

[36] Khae Duc Do, Zhong-Ping Jiang, and J. Pan. Robust adaptive path following of underactuated ships. *Automatica*, 40(6):929–944, 2004.

[37] Alexander Domahidi, Eric Chu, and Stephen Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[38] Jingyan Dong, Placid Ferreira, and James A. Stori. Feed-rate optimization with jerk constraints for generating minimum-time trajectories. *International Journal of Machine Tools and Manufacture*, 47(12):1941–1955, 2007.

[39] Pedro Encarnaçao and António Pascoal. Combined trajectory tracking and path following: an application to the coordinated control of autonomous marine craft. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 964–969. IEEE, 2001.

[40] Timm Faulwasser and Rolf Findeisen. Nonlinear model predictive control for constrained output path following. *IEEE Transactions on Automatic Control*, 61(4):1026–1039, 2016.

[41] Timm Faulwasser, Tobias Weber, Pablo Zometa, and Rolf Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2017.

[42] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, dec 2014. ISSN 1867-2949. doi: 10.1007/s12532-014-0071-1. URL http://link.springer.com/10.1007/s12532-014-0071-1.

[43] Francisco Geu Flores and Andrés Kecskeméthy. Time-optimal path planning for the general waiter motion problem. In Vijay Kumar, James Schmiedeler, S V Sreenivasan, and Hai-Jun Su, editors, *Mechanisms and Machine Science*, volume 14, pages 189–203. Springer International Publishing, Heidelberg, 2013. ISBN 978-3-319-00397-9. doi: 10.1007/978-3-319-00398-6_14. URL https://doi.org/10.1007/978-3-319-00398-6_14.

[44] Komei Fukuda and Alain Prodon. Double description method revisited. In *Combinatorics and computer science*, pages 91–111. Springer, 1996.

[45] Christopher G. Atkeson, Benzun Pious Wisely Babu, Nandan Banerjee, Dmitry Berenson, Christoper P. Bove, Xiongyi Cui, Mathew Dedonato, Ruixiang Du, Siyuan Feng, Perry Franklin, M Gennert, Joshua P. Graff, Peng He, Aaron Jaeger, Joohyung Kim, Kevin Knoedler, Lening Li, Chenggang Liu, Xianchao Long, and X Xinjilefu. *What Happened at the DARPA Robotics Challenge Finals*, pages 667–684. Springer, 2018. ISBN 978-3-319-74665-4. doi: 10.1007/978-3-319-74666-1_17.

[46] Alessandro Gasparetto and V. Zanotto. A new method for smooth trajectory planning of robot manipulators. *Mechanism and machine theory*, 42(4):455–471, 2007.

[47] Alessandro Gasparetto and V. Zanotto. A technique for time-jerk optimal planning of robot trajectories. *Robotics and Computer-Integrated Manufacturing*, 24(3):415–426, 2008.

[48] Reinhard Geissbauer, Evelyn Lubben, Stefan Schrauf, and Steve Pillsbury. Digital champions & how industry leaders build integrated operations ecosystems to deliver end-to-end customer solutions. Technical report, PwC, 2018.

[49] Mitsuo Gen and Runwei Cheng. *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons, 2000.

[50] Oscar Gerelli and Corrado Guarino Lo Bianco. Real-time Path Tracking Control of Robotic Manipulators with Bounded Torques and Torque-derivatives. *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2008*, 2008.

[51] Oscar Gerelli and Corrado Guarino Lo Bianco. Nonlinear variable structure filter for the online trajectory scaling. *IEEE Transactions on Industrial Electronics*, 56(10):3921–3930, 2009. ISSN 02780046. doi: 10.1109/TIE.2009.2018431.

[52] Oscar Gerelli and Corrado Guarino Lo Bianco. A discrete-time filter for the on-line generation of trajectories with bounded velocity, acceleration, and jerk. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3989–3994, 2010. ISSN 10504729. doi: 10.1109/ROBOT.2010.5509712.

[53] Kris Hauser. Fast Interpolation and Time-Optimization on Implicit Contact Submanifolds. In *Robotics: Science and Systems*. Citeseer, 2013.

[54] Kris Hauser. Fast interpolation and time-optimization with contact. *The International Journal of Robotics Research*, 33(9):1231–1250, aug 2014. ISSN 0278-3649. doi: 10.1177/0278364914527855. URL http://journals.sagepub.com/doi/10.1177/0278364914527855.

[55] Kris Hauser. Fast dynamic optimization of robot paths under actuator limits and frictional contact. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2990–2996. IEEE, 2014.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[57] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of Honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1321–1326. IEEE, 1998.

[58] Arjang Hourtash. The Kinematic Hessian and Higher Derivatives. *2005 International Symposium on Computational Intelligence in Robotics and Automation*, 2005. doi: 10.1109/CIRA.2005.1554272.

[59] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44. IEEE, 2016.

[60] International Federation of Robotics. World Robotics 2018 Industrial Robots. Technical report, International Federation of Robotics, 2018. URL https://ifr.org/downloads/press2018/Executive{_}Summary{_}WR{_}2018{_}Industrial{_}Robots.pdf.

[61] 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.

[62] Dominik Kaserer, Hubert Gattringer, and Andreas Müller. Nearly Optimal Path Following With Jerk and Torque Rate Limits Using Dynamic Programming. *IEEE Transactions on Robotics*, 2018.

[63] Eric Colin Kerrigan. *Robust constraint satisfaction: Invariant sets and predictive control*. PhD thesis, University of Cambridge, 2001.

[64] Jon Kieffer, Aidan J. Cahill, and Matthew R. James. Robust and accurate time-optimal path-tracking control for robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(6):880–890, 1997. ISSN 1042296X. doi: 10.1109/70.650167.

[65] James Kuffner and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[66] Tobias Kunz and Mike Stilman. Time-optimal trajectory generation for path following with bounded acceleration and velocity. *Robotics: Science and Systems VIII*, 2012.

[67] Kostas J Kyriakopoulos and George N Saridis. Minimum jerk path generation. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 364–369. IEEE, 1988.

[68] P. M. Larochelle, J. Michael McCarthy, and James E. Bobrow. Determining maximum payloads for cooperating robots under time-optimal control. *Robotics and computer-integrated manufacturing*, 10(6):437–443, 1993.

[69] Steven M. LaValle. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. doi: 10.1177/02783640122067453. URL http://ijr.sagepub.com/content/20/5/378.

[70] Puttichai Lertkultanon and Quang-Cuong Pham. Dynamic non-prehensile object transportation. In *2014 13th International Conference on Control Automation Robotics and Vision, ICARCV 2014*, pages 1392–1397. IEEE, 2014. ISBN 9781479951994. doi: 10.1109/ICARCV.2014.7064519.

[71] Sergey Levine and Pieter Abbeel. Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics. In *Advances in Neural Information Processing Systems*, pages 1–3, 2014. ISBN 9781479969227. doi: 10.1109/ICRA.2015.7138994.

[72] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. ISSN 15337928. doi: 10.1007/s13398-014-0173-7.2.

[73] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *arXiv preprint arXiv:1603.02199*, 2016. ISSN 0278-3649. doi: 10.1177/0278364917710318. URL http://arxiv.org/abs/1603.02199.

[74] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2011. ISBN 9788578110796. doi: 10.1017/CBO9781107415324.004.

[75] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

[76] Liang Liu, Chaoying Chen, Xinhua Zhao, and Yangmin Li. Smooth trajectory planning for a parallel manipulator with joint friction and jerk constraints.

International Journal of Control, Automation and Systems, 14(4):1022–1036, 2016.

[77] T. Lozano-Pérez. Robot programming. *Proceedings of the IEEE*, 71(7):821–841, 1983.

[78] Jingru Luo and Kris Hauser. Robust trajectory optimization under frictional contact with iterative learning. *Autonomous Robots*, 41(6):1447–1461, 2017. ISSN 15737527. doi: 10.1007/s10514-017-9629-x.

[79] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Planning, and Control*. Cambridge University Press, 2017. ISBN 9781107156302.

[80] Ian R. Manchester and Jean-Jacques E. Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 2017.

[81] Humberto Martínez-Barberá and David Herrero-Pérez. Autonomous navigation of an automated guided vehicle in industrial environments. *Robotics and Computer-Integrated Manufacturing*, 26(4):296–311, 2010.

[82] Jan Mattmüller and Damian Gisler. Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path. *The International Journal of Advanced Manufacturing Technology*, 45(9-10):1007–1016, 2009.

[83] Iñaki Maurtua, Aitor Ibarguren, Johan Kildal, Loreto Susperregi, and Basilio Sierra. Human–robot collaboration in industrial applications: Safety, interaction and trust. *International Journal of Advanced Robotic Systems*, 14(4): 1729881417716010, 2017.

[84] J. Michael McCarthy and James E. Bobrow. The number of saturated actuators and constraint forces during time-optimal movement of a general robotic system. *IEEE Transactions on Robotics and Automation*, 8(3):407–409, 1992.

[85] Douglas Morrison, Adam W Tow, M McTaggart, R Smith, N Kelly-Boxall, S Wade-McCue, J Erskine, R Grinover, A Gurman, and T Hunn. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. *arXiv preprint arXiv:1709.06283*, 2017.

[86] Marco Muenchhof and Tarunraj Singh. Jerk limited time optimal control of flexible structures. *Transactions-American Society of Mechanical Engineers Journal of Dynamic Systems Measurement and Control*, 125(1):139–142, 2003.

[87] Michael P Murphy, Aaron Saunders, Cassie Moreira, Alfred A Rizzi, and Marc H Raibert. The littledog robot. *The International Journal of Robotics Research*, 30(2):145–149, 2011.

[88] Ákos Nagy and István Vajk. LP-based velocity profile generation for robotic manipulators. *International Journal of Control*, 91(3):582–592, 2018.

[89] Yoshihiko Nakamura and Hideo Hanafusa. Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163, 1986. ISSN 00220434. doi: 10.1115/1.3143764.

[90] Christopher Nielsen, Cameron Fulford, and Manfredi Maggiore. Path following using transverse feedback linearization: Application to a maglev positioning system. *Automatica*, 46(3):585–590, 2010.

[91] Matthias Oberherber, Hubert Gattringer, and Andreas Müller. Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking. *Mechanical Sciences*, 6(2):245–254, 2015.

[92] Katsuhiko Ogata and Yanjuan Yang. *Modern control engineering*, volume 4. Prentice hall India, 2002.

[93] J-S Pang and J Trinkle. Stability characterizations of rigid body contact problems with coulomb friction. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 80(10):643–663, 2000.

[94] Friedrich Pfeiffer and Rainer Johanni. A concept for manipulator trajectory planning. *Robotics and Automation, IEEE Journal of*, 3(2):115–123, 1987.

[95] Hung Pham and Quang-Cuong Pham. On the Structure of the Time-Optimal Path Parameterization Problem with Third-Order Constraints. In Allison M. Okamura, editor, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017. IEEE. ISBN 9781509046324. URL http://arxiv.org/abs/1609.05307.

[96] Hung Pham and Quang-Cuong Pham. A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. *IEEE Transactions on Robotics*, jul 2018. ISSN 15523098. doi: 10.1109/TRO.2018.2819195. URL https://arxiv.org/abs/1707.07239.

[97] Hung Pham and Quang-Cuong Pham. Time-Optimal Path Tracking via Reachability Analysis. In *2018 IEEE International Conference on Robotics and Automation*, 2018. ISBN 9781538630808. URL http://arxiv.org/abs/1709.05101.

[98] Hung Pham and Quang-Cuong Pham. Critically fast pick-and-place with suction cups. *Accepted at 2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[99] Quang-Cuong Pham. A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm. *IEEE Transactions on Robotics*, 30 (6):1533–1540, dec 2014. ISSN 1552-3098. doi: 10.1109/TRO.2014.2351113. URL http://ieeexplore.ieee.org/document/6895310/.

[100] Quang-Cuong Pham and Olivier Stasse. Time-Optimal Path Parameterization for Redundantly Actuated Robots: A Numerical Integration Approach. *IEEE/ASME Transactions on Mechatronics*, 20(6):3257–3263, dec 2015. ISSN 1083-4435. doi: 10.1109/TMECH.2015.2409479. URL http://ieeexplore.ieee.org/document/7083769/.

[101] Quang-Cuong Pham, Stéphane Caron, Puttichai Lertkultanon, and Yoshihiko Nakamura. Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots. *The International Journal of Robotics Research*, page 0278364916675419, 2016.

[102] Aurelio Piazzi and Antonio Visioli. Global minimum-jerk trajectory planning of robot manipulators. *IEEE transactions on industrial electronics*, 47(1):140–149, 2000.

[103] Ionela Prodan, Sorin Olaru, Fernando ACC Fontes, Cristina Stoica, and Silviu-Iulian Niculescu. A predictive control-based algorithm for path following of autonomous aerial vehicles. In *Control Applications (CCA), 2013 IEEE International Conference on*, pages 1042–1047. IEEE, 2013.

[104] Sasa V. Rakovic, Eric Colin Kerrigan, David Q. Mayne, and John Lygeros. Reachability analysis of discrete-time systems with disturbances. *IEEE Transactions on Automatic Control*, 51(4):546–561, 2006.

[105] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[106] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.

[107] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. *Robotics: Science and Systems (RSS)*, 2013.

[108] Raimund Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6(3):423–434, 1991.

[109] Zvi Shiller. Time-energy optimal control of articulated systems with geometric path constraints. *Journal of dynamic systems, measurement, and control*, 118 (1):139–143, 1996.

[110] Zvi Shiller and Steven 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.

[111] Zvi Shiller and Y-R Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249, 1991.

[112] 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, 1992.

[113] Kang Shin and Neil Davis McKay. Minimum-time control of robotic manipulators with geometric path constraints. *Automatic Control, IEEE Transactions on*, 30(6):531–541, 1985.

[114] Kang Shin and Neil Davis McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, 31(6):491–500, 1986.

[115] Kang Shin and Neil Davis McKay. Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automatic Control*, 31 (6):501–511, 1986.

[116] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[117] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[118] Arun Kumar Singh and K Madhava Krishna. A class of non-linear time scaling functions for smooth time optimal control along specified paths. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5809–5816. IEEE, 2015.

[119] Jean-Jacques E. Slotine and Hyun S. Yang. Improving the efficiency of time-optimal path-following algorithms. *Robotics and Automation, IEEE Transactions on*, 5(1):118–124, 1989.

[120] Mark W. Spong and Mathukumalli Vidyasagar. Robot dynamics and control, 2008.

[121] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 2012.

[122] Josef Stoer and Roland Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

[123] R S Sutton and A G Barto. Reinforcement learning: an introduction., 1998. ISSN 1045-9227.

[124] Mikko Tarkiainen and Zvi Shiller. Time optimal motions of manipulators with actuator dynamics. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 725–730. IEEE, 1993.

[125] D. Verscheure, B. 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*, 2008.

[126] Iris FA Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006.

[127] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373, 1992.

[128] Miomir Vukobratović and Branislav Borovac. Zero-Moment Point — Thirty Five Years of Its Life. *International Journal of Humanoid Robotics*, 01(01): 157–173, 2004. ISSN 0219-8436. doi: 10.1142/S0219843604000083.

[129] Nick Wingfield. As Amazon Pushes Forward With Robots, Workers Find New Roles, 2017. URL https://www.nytimes.com/2017/09/10/technology/amazon-robots-workers.html.

[130] Qiang Zhang, Shu-Rong Li, and Xiao-Shan Gao. Practical smooth minimum time trajectory planning for path following robotic manipulators. In *American Control Conference (ACC), 2013*, pages 2778–2783. IEEE, 2013.

[131] Leon Zlajpah. On time optimal path control of manipulators with bounded joint velocities and torques. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1572–1577. IEEE, 1996.

[132] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.